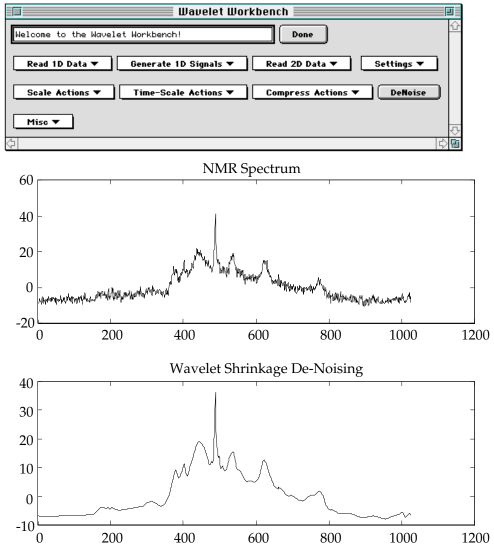
The IDL Wavelet Workbench

Research Systems, Inc.

Amara Graps



2995 Wilderness Place, Suite 203, Boulder, CO 80301, USA, TEL +1-303-786-9900, FAX +1-303-786-9909

Wavelets are functions that satisfy certain mathematical requirements and are used in representing data or other functions. This idea is not new. Approximation using superposition of functions has existed since the early 1800's, when Joseph discovered that he could superpose sines and cosines to represent other functions. However, in wavelet analysis, the *scale* that we use to look at data plays a special role. Wavelet algorithms process data at different *scales* or *resolutions*. If we look at a signal with a large "window," we would notice gross features. Similarly, if we look at a signal with a small "window," we would notice small features. The result in wavelet analysis is to see both the forest *and* the trees, so to speak.

This makes wavelets very useful. For many decades, scientists have wanted more appropriate functions than the sines and cosines which comprise the bases of Fourier analysis, to approximate choppy signals. By their definition, these Fourier functions are non-local, and therefore do a very poor job in approximating sharp spikes. But with wavelet analysis, we can use approximating functions that are contained neatly in finite domains. Wavelets are well-suited for approximating data with sharp discontinuities.

The wavelet analysis procedure is to adopt a wavelet prototype function, called an *analyzing wavelet* or *mother wavelet*. Temporal analysis is performed with a contracted, high-frequency version of the prototype wavelet, while frequency analysis is performed with a dilated, low-frequency version of the same wavelet. Because the original signal or function can be represented in terms of a wavelet expansion (using coefficients in a linear combination of the wavelet functions), data operations can be performed using just the corresponding wavelet coefficients. And if you further choose the best wavelets adapted to your data, or truncate the coefficients below a threshold, your data is sparsely represented. This sparse coding makes wavelets an excellent tool in the field of data compression.

Other applied fields that are making use of wavelets include astronomy, acoustics, nuclear engineering, sub-band coding, signal and image processing, neurophysiology, music, magnetic resonance imaging, speech discrimination, optics, fractals, turbulence, earthquake-prediction, radar, human vision, and pure mathematics applications such as solving partial differential equations.

1. Installation

The Wavelet Workbench is a library of about 150 IDL procedures, functions, and programs that can be most easily run from the Graphic User Interface module called: wwb.pro. The routines can be run from the IDL command line as well, if the data and the necessary inputs are spectified.

The Installation procedure is as follows.

1. Log onto `gateway.rsinc.com` via anononymous FTP.
2. **cd** to the directory: `/incoming`.
3. FTP the `wwb` file, compressed for your platform, in binary mode.
4. Uncompress the wwb file into a directory using the appropriate software for your platform.
5. Copy that directory to a desired location on your machine.

2. Starting the Program

The `wwb.pro` procedure is the IDL widget designed for the easiest access to all of the Wavelet Workbench functions. This widget contains pull-down menus to select the input data and perform wavelet functions. The output is to standard IDL windows.

1. Start up IDL.
2. CD to the directory containing Wavelet Workbench.
   (Note: On the Macintosh, you would instead "Set Working Directory" under the **File** menu.)
3. Type `wwb`.
4. You'll see a Wavelet Workbench widget. Now you can begin working with the program.

1. Set the Wavelet Workbench Directory

The Wavelet Workbench routines contain several directories of data files. In order for the main routine: `wwb.pro` to find these data files, please set the path for the Wavelet Workbench directory.

You have two ways of setting the path:

- Edit the file `wwb.pro` to change the default directory.
- Type in a different path when you see the directory widget.

To edit the `wwb.pro` file, look for the section in the file:

```
_________CHANGE WWB DEFAULT DIRECTORY HERE_________
;Change the default string to reflect the location of the WWB folder:
default_str = 'Mac Hard Disk:wwb:'
```

and edit the `default_str` string to be the path at which your WWB directory is located. Then, when the default directory widget appears at the startup of Wavelet Workbench, press **OK**.

To type in a different path when you see the directory widget, simply type in your new path in the edit box presented to you. Then, press **OK**.

2. Select the Data or Signal

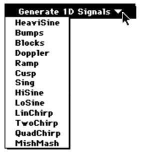
To use the Wavelet Workbench, you have to give it some data. You can give it either:

- A *pre-defined* 1D dataset.
- A *user-defined* 1D dataset.
- A pre-defined 1D *signal*.
- A *pre-defined* 2D dataset (image).
- A *user-defined* 2D dataset (image).

2.1. Select a Pre-defined or User-defined 1D Dataset or Signal

To select a *pre-defined* 1D dataset, all you need to do is go to the **Read 1D Data** menu item, and select one of the choices.

To select a *pre-defined* 1D signal, all you need to do is go to the **Generate 1D Signals** menu item, and select one of the choices.



Setting Up a User-Defined 1D Dataset

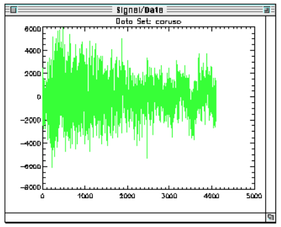
To select a *user-defined* 1D dataset, you have to do some preparatory work first. Create a text file called `xx.doc`, where `xx` is a meaningful name for your data. For the first line of the file, place an integer value that represents the number of elements (number of rows) that your data file contains, for example: 1024. Then, from the second line on down, write a description of the data file. Here's one example of a 'doc' file called `tweet.doc`:

```
2048
tweet.asc -- Digital signal of a bird singing

Access
    read_data, 'tweet', 2048

Size
    2048

Sampling Rate
    22 kHz

Source
    Doug Jones, University of Illinois
```

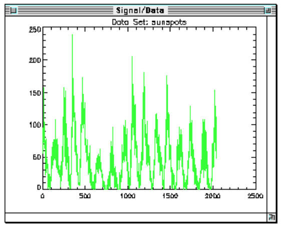

In addition to the `xx.doc` file, you'll need to create the data file itself. Wavelet Workbench will read in ASCII-type data files of one value per row. Name the data file `xx.asc`, where `xx` is the same name as the corresponding `xx.doc` file. Now place these two files in the directory titled: `u1data`. When Wavelet Workbench starts, it will look in this directory and create a menu item for each file with a `xx.asc` name. Then select the dataset as if it were one of the pre-defined datasets described previously.

Wavelet Workbench currently offers the following 1D datasets and signals. For 1D datasets, we have:

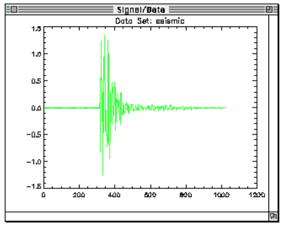
- Caruso- A digital signal of Caruso singing.



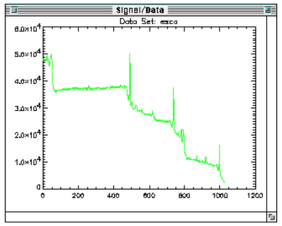
- Sunspots- Monthly sunspot numbers provided by J. Scargle, NASA-Ames.



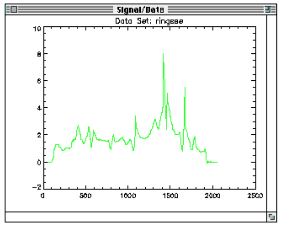
- Seismic- A seismic signal, that is distributed throughout the seismic industry as a test dataset.



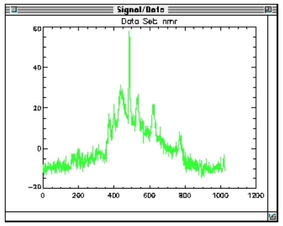
- ESCA- An ESCA (Electron Spectroscopy for Chemical Analysis) spectrum.



- Ring- A Voyager spacecraft Photopolarimeter stellar occultation rings dataset.

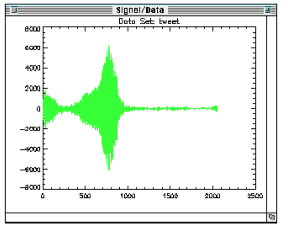


- NMR- A Nuclear Magnetic Resonance signal.



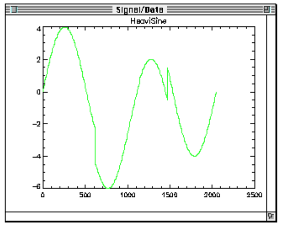
An additional 1D dataset from the `u1ddata` folder is:

- tweet- A digital signal of a bird singing.

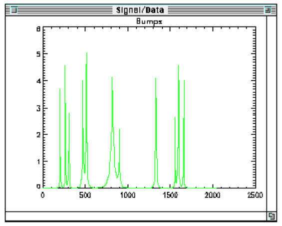


For 1D signals Wavelet Workbench offers:

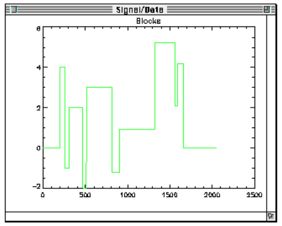
- HeaviSine- A sinusoid with two "clicks."



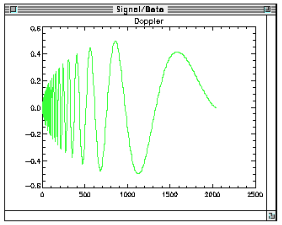
- Bumps- A very "peaky" signal, like an NMR spectrum.



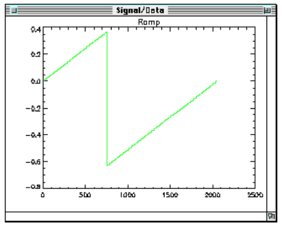
- Blocks- A piecewise-constant signal, like a line-scan through a 2D image.



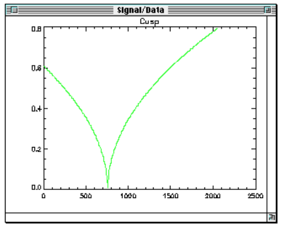
- Doppler- An attenuating sinusoid with time-varying frequency.



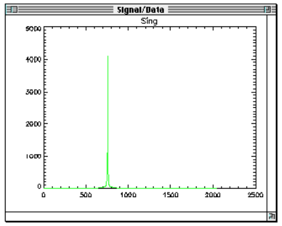
- Ramp- A ramp.



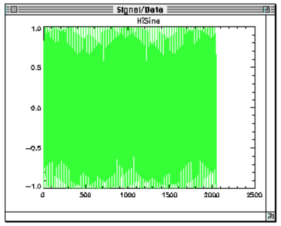
- Cusp- A cusp.



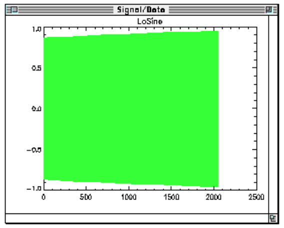
- Sing- A singularity.



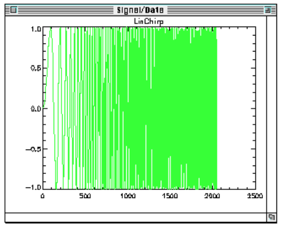
- HiSine- A high-frequency sinusoidal signal.



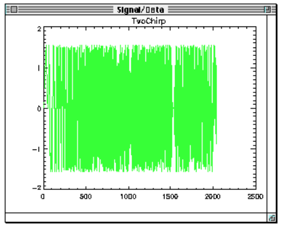
- LoSine- A low-frequency sinusoidal signal.



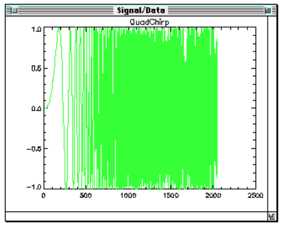
- LinChirp- A linear chirp signal.



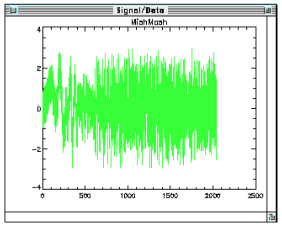
- TwoChirp- A sum of two linear chirp signals.



- QuadChirp- A quadratic chirp signal.



- MishMash- A superposition of QuadChirp with a linear chirp and a high-frequency sinusoid.



2.2. Select a Pre-Defined or User-Defined 2D Dataset

To select a *pre-defined* 2D dataset, all you do is go to the **Read 2D Data** menu item and select one of the choices.

Setting Up a User-Defined 2D Dataset

To select a *user-defined* 2D dataset, you have to do some preparatory work first, in exactly the same fashion as for the previous user-defined 1D dataset. Create a text file called `xx.doc`, where `xx` is a meaningful name for your data. For the first line of the file, place an integer value that represents the number of elements in a row (Wavelet Workbench assumes *square* images) that your data file contains. Then from the second line on down, write a description of the data file. Here's one example of a 'doc' file called `phone.doc`:

```
256
phone.raw -- Gray-scale image of telephone

Access
    READ_IMAGE, 'phone', 256

Size
    256 by 256

Gray Levels
    256

Description
    Images liks this are used often in the edge
    detection literature of computer vision.
```

In addition to the `xx.doc` file, you'll need to create the data file itself. Name the data file `xx.raw`, where `xx` is the same name as the corresponding `xx.doc` file. Now place these two files in the directory titled: `u2ddata`. When Wavelet Workbench starts up, it will look in this directory and create a menu item for each file with a `xx.asc` name. Then, select the dataset as if it were one of the pre-defined datasets described previously.

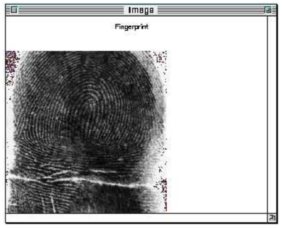
Wavelet Workbench will read in byte type data files of the sort that IDL produces when writing to a byte file via **openw** and **assoc**. For example, the following IDL code reads in a picture of a phone, flips it sideways, scales it to be a larger 256 by 256 size image (instead of 128 by 128), then writes it out to a byte file. The type of file written in this manner will be readable by Wavelet Workbench using its `wreadimg.pro` procedure.


```
;open the data file and read it
get_lun, lun
openr, lun, 'phone.raw'
len = 128
a = assoc(lun, bytarr(len, len))
image = a(0)
close, lun
free_lun, lun

;Rotate so that it's not sideways
r = rotate(image, 3)
tvscl, r

;Expand to encompass 256 by 256
t = rebin(r, 256, 256)

;Now write to a new file and close/free logical units
get_lun, lun
openw, lun, 'phone2.raw'
b = assoc(lun, bytarr(256, 256))
b(0) = t
close, lun
free_lun, lun
end
```

Wavelet Workbench currently offers the following 2D datasets:

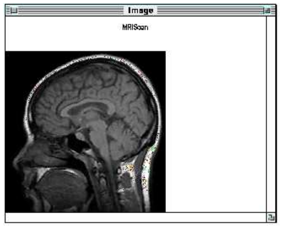
- Daubechies- An Image of Ingrid Daubechies (an active wavelet researcher).

- Fingerprint- A gray scale fingerprint image from the FBI's database.



- Lincoln- A gray scale image of Abe Lincoln.

- MRIScan- A gray-scale MRI brain scan.



An additional 2D dataset from the `u2ddata` folder is:

- phone- A gray-scale image of telephone, useful for edge-detection.

3. Select Parameter Settings

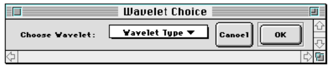
The **Settings** menu are parameters that need to be set in order for the Wavelet Workbench to perform its calculations.

However, default values are set, so if you choose "actions" without going through this menu, you won't have any problems.

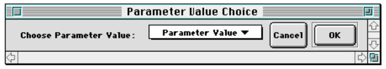
3.1. Wavelet

This menu item selects the wavelet, or more precisely the *quadrature mirror filters* (QMF) for all of the wavelet operations. You have four choices of wavelets: Haar, Daubechies, Coiflet, and Symmlet.

Each of these wavelets has an associated integer parameter which defines the sub-member of that family. Mathematically, the integer is related to the number of vanishing moments. The procedure is to select a wavelet, and when you see a second widget for which to choose a parameter value, select one of those parameters. For both the wavelet choice:



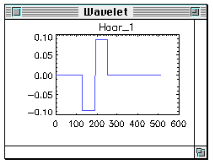
and the parameter choice,



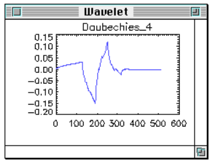
a widget will pop up for you to select parameters from a menu.

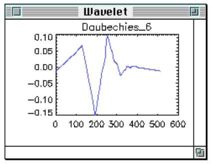
You can see what the wavelet looks like each time you choose a wavelet family and parameter value, because a plot window with the wavelet you specfied will appear after you've made your selection from the menus. The parameter choices for each wavelet are the following:

- Haar: 1

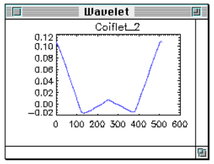


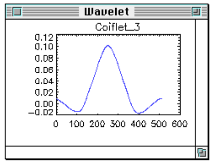
- Daubechies: 4,6



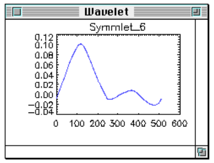


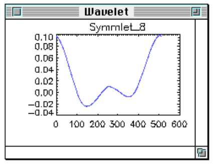
- Coiflet: 2,3





- Symmlet: 6,8





The following is an example of choosing a wavelet.

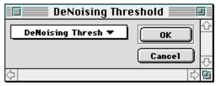
1. Select the **Wavelet** menu item.
2. Select **Daubechies.** Then press **OK**
3. Choose Parameter **4.** Then press **OK**
4. You'll see the wavelet plotted in a little window.

The Default is the **Haar** wavelet.

3.2. DeNoising Threshold

This parameter is necessary for the **DeNoise** action, i.e. the calculations that are performed when you press the **DeNoise** button. Your choices of methods are particular statistical techniques developed by researchers in the wavelets field.

The procedure is the following.



1. Select **DeNoising Threshold.**
2. Select one of the four methods below. Then, press **OK.**
   - VisuThresh
   - MultiHybrid
   - MultiMAD
   - MinMaxThresh
   - WPDeNoise

Here is a brief summary of each of these methods.

VisuThresh = Visually calibrated adaptive smoothing method.

MultiHybrid = Stein's Unbiased Risk Estimate Variant without Rigorous Proof method.

MultiMAD = Level-dependent Noise level estimation so that the "MAD" of the coefficient at each level = .6745

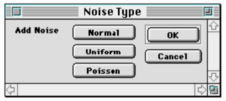
MinMaxThresh = Minimax Thresholding method.

WPDeNoise = Wavelet Packet DeNoising method. This method is very slow! (I don't recommend this method.)

The Default is the **VisuThresh** method.

3.3. Noise Type

This parameter is necessary for the **Misc- Adding Noise** action.



Your choices of **Noise Type** are:

- Normal
- Uniform
- Poisson

Each of these choices (Normal, Uniform, or Poisson) are mathematical distributions that the noise variances will be drawn from.

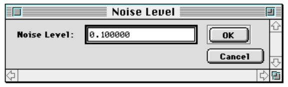
The following is an example of choosing a **Noise Type.**

1. Select the **Noise Type** menu.
2. Select **Uniform.** Then press **OK**

The Default is the **Normal** distribution.

3.4. Noise Level

This parameter is necessary for the **Misc- Adding Noise** menu item action.



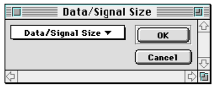
Your choice of **Noise Level** is whatever you wish to type into the edit box. This value reflects the amplitude of the noise added. The default is 0.1, which is a good number for 1D generated signals. For 2D image data, try setting this number to 4.0 or bigger to see a difference.

The following is an example of setting the **Noise Level.**

1. Select the **Noise Level** menu.
2. Type **5.0** into the edit box. Then, press **OK.**

3.5. Data/Signal Size

This parameter is necessary for setting the *length* of the 1D Signals/Datasets, or the *size* of 2D Datasets.



When you see the widget, select **64, 128, 256, 512, 1024,** or **2048** as the size. The default is 1024 for 1D generated signals. The real 1D/2D datasets have their length determined automatically when they are first read in. The size can then be reduced using this function.

The following is an example of setting the **Data/Signal Size.**

1. Select the **Data/Signal Size** menu.
2. Choose **512** from the menu selection. Then press **OK.**

4. Scale Action

The action items under this menu are simple wavelet actions.

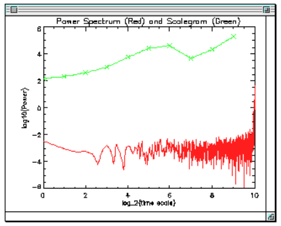
4.1. Scalegram

The scalegram is the mean square wavelet transform coefficient, averaged over location, i.e. over the dyad of the dataset. So, for an example of a 2048-value-long dataset, the last 1024 wavelet transform coefficient points are averaged for those points, and represented by an 'x.' Then the next dyad from the end, which would be 512 points in this example, are averaged, and represented by an 'x,' and so on. Currently, the only wavelet being used for Scalegrams is the Haar wavelet, and this function works just on 1D datasets.

Here is an example of creating a scalegram.

1. Select the **Scalegram** menu.
2. See a new window appear with the power spectrum (red) and the scalegram (green) of your dataset.

For example, a scalegram of the Sunspots 1D dataset will give you the following output:



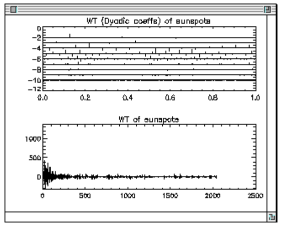
The scalegram operation works on 1D datasets, but not 2D datasets.

4.2. Discrete Wavelet Transform

The Discrete Wavelet Transform (DWT) is probably the key component of this software package. It is the analog of the Discrete Fourier Transform, except, here, you're using wavelet bases instead of sine and cosine bases. When you select this menu item, you'll see the frequency components of the dataset for each scale. The input dataset (whether it is 1D or 2D) will determine how the output frequency components will look. This issue I discuss next.

For a 1D Dataset/Signal

When selecting **DWT** for a 1D dataset/signal, you will see a plot split into two panes- an upper pane and a lower pane. The lower pane plot shows the components for the whole input dataset, but at different scales. The upper dyadic plot shows the coefficients for each of the dyad-scales of the data. So, for the finest scale, the latter dyad (points: 1024 to 2048 of a 2048-value-long dataset) contain the coefficients for the frequency. One level down is the next dyad-scale (representing points 512-1024) of the wavelet transform coefficient array, and so on. The next figure shows a discrete wavelet transform of the Sunspots dataset.

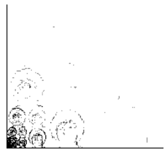


To perform a discrete wavelet transform on 1D data, take the following steps.

1. Select **DWT** from the **Scale Actions** menu.
2. See a new window appear with two plots of your transform coefficients.

For a 2D Dataset

When selecting **DWT** for a 2D dataset (image), The DWT decomposes the original image into an image of transform coefficients. Select a **Read 2D Dataset,** say the **Daubechies** image. Now select **DWT.** The output coefficient matrix size is same is input size of matrix (square, length 2^n on each side, so a 64x64, or 128x128, or 256x256 matrix etc). The original image is scaled and smoothed such that the lower left square shows the smallest scaled version of the original, and the other squares show different detailed and smoothed versions. The next figure shows a discrete wavelet transform of the Daubechies 2D dataset.



The DWT operates in the following way. For each scale, it performs a down-sample, high-pass smoothing operation on half of the data, then performs a down-sample, low-pass operation (throw away every second data point) of the other half of the data. The result is a square with squares-within-squares of low-pass (L) operations and high-pass operations (H). The filter operations always work in pairs (*Quadrature Mirror Filters*), the filters being the wavelets you've chosen (say the Daubechies wavelet).

Next is an ASCII sketch of some output filter operations. The sketch shows three scale levels of smoothing, down-sampled operations. The square labelled all *HH* will be the most bland, smoothed result. The square with all *LL* will be the tiniest representation of your original image (just down-sampled, with every other point thrown away several times). The others will show you different things. Some combinations of LH will show the fine detail in your image. Other combinations will show the sharp contrast portions of your image (those are the combinations that make wavelets good *edge detectors*).

```
-----------------------------------
|                |                |
|                |                |
|                |                |
|                |                |
|                |                |
|      L,H       |       H,H      |
|                |                |
|                |                |
|                |                |
|                |                |
-----------------------------------
|        |        |               |
| LL,LH  |LH,LH   |               |
|        |        |               |
|        |        |               |
|        |        |               |
|-----------------|     H,L       |
|LLL|LLH,|        |               |
|LLH|LLH |        |               |
|--------| LH,LL  |               |
|LLL|LLH,|        |               |
|LLL|LLL |        |               |
-----------------------------------
```

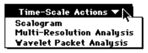
To perform a discrete wavelet transform on 2D data, take the following steps.

1. Select **DWT** from the **Scale Actions** menu.
2. See a new window appear with 2D image of your transform coefficients.

This computation is very intensive, and will take a few minutes, so perhaps you should get a cup of coffee while it's calculating! One workaround the long computation time is to rescale the size of the image into a smaller image by using the **Data/Signal Size** of the **Settings** menu item.

5. Time-Scale Action

The action items under this menu item are more complex wavelet actions.



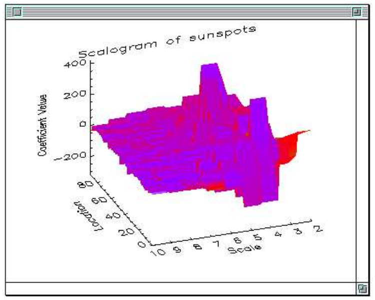
5.1. Scalogram

The scalegram is a time-scale representation of the signal, and is the discrete wavelet transform coefficients plotted as a function of the indices: *scale* and *location*. The representation in Wavelet Workbench is a surface plot. The scalogram may be useful for functions such as feature extraction, parameter estimation and pattern recognition for nonstationary signals because abrupt changes in a signal invariably produce high frequency components. Therefore, at small scales, which correspond to high frequency components, the scalogram will have high magnitudes.

Here is an example of creating a scalogram in Wavelet Workbench.

1. Select **Scalogram.**
2. See a new window appear with a surface plot scalogram of your dataset.

The next figure shows a scalogram of the Sunspots 1D dataset.



The scalogram operation works on 1D datasets, but not currently on 2D datasets.

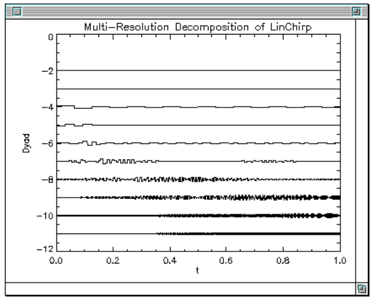
5.2. Multi-resolution Analysis

A wavelet Multi-resolution Analysis (MRA) decomposes a signal, using the selected wavelet as a filter, into components of different scales of 2^{-m}, m, an integer. The wavelet chosen, or *QMF*, serves as a low pass and high pass filter, which separates the signal into detail components plus a coarse approximation. Then the output is subsampled again. The coarsest resolution is the top level.

Here is an example of performing a multi-resolution analysis on a dataset.

1. Select **Multi-Resolution Analysis.**
2. See a new window appear with multi-resolution plot of your dataset.

The following psfile=shows a multi-resolution analysis of the linear chirp signal using the Haar wavelet.



The multi-resolution operation works on 1D datasets, but not currently on 2D datasets.

5.3. Wavelet Packet Analysis

Wavelet Packet Analysis decomposes the signal into its scale and frequency components, then looks for combinations of large components, or of not-so-large components that share similar time or frequency location. When a large component is found, the time-frequency location is marked and used in building a time-frequency *tree* of the analyzed signal. The *best basis* is calculated from the time-frequency tree.

There are some similarities with the multi-resolution analysis in that the input signal is split – one of the bands becomes one of the outputs, while the lower band is again split. But instead of splitting into two bands, as the MRA does, the wavelet packet analysis performs a more general analysis, so that several splits can occur in both the high and lower bands.

Three displays are shown for the wavelet packet analysis. The first is the `Wavelet Packet Decomposition` plot.

The Wavelet Packet Decomposition Display

This first plot is the original signal/dataset at:

1. different scalings (*downsampled by 2,* i.e. every other point is kept), and
2. sampled for high and low frequency components.

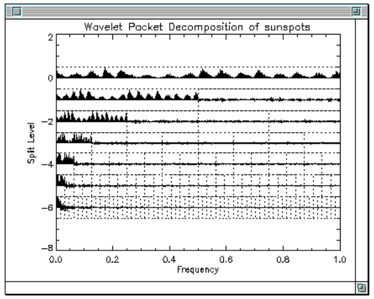
The scalings are labelled as *Split Levels,* and shown on the y-axis. The levels are recursive in that each level from the top down are analyzed with low and high pass filters of the data in the level above it.

The high and low pass filters analyze the data shown in the level above it. The filtering process alternates the low and high frequency components horizontally. Here's how it works.

```
Split Level ``0"    Original Data Set of Length N.

Split Level ``1".   Level 0 data is analyzed.
Low Freq Components| High Freq Components
0                  N/2                  N

Split Level ``2". Level 1 data is analyzed.
0 ->  N/2  | N/2 -> N
Low   High | High   Low
0   N/4    N/2    3N/4   N


Split Level ``3". Level 2 data is analyzed.
0 ->  N/4   | N/4 -> N/2 | N/2 ->  3N/4  | 3N/4 -> N
Low   High   High  Low    Low     High    High    Low
0   N/8    N/4    3N/8  N/2   5N/8     3N/4    7N/8   N
```

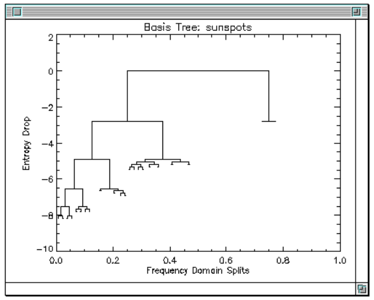
And so on. Because a successive series of high and low pass downsampling operations take place on the original data set shown at the top, the lowest split level is essentially *frequency* taken to the limit.

The psfile=on the next page shows the wavelet packet decomposition for the Sunspots dataset.



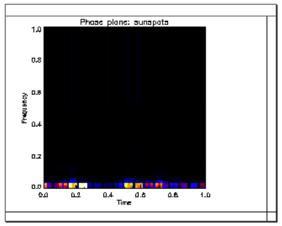
The Entropy Drop Tree Display

After the wavelet packet decomposition, a best basis is calculated, which chooses the best basis for your dataset. The next display in the wavelet packet analysis is the `Entropy Drop Tree`. The best basis algorithm finds the coefficient sequence in the root subspace, then follows the branches of the wavelet packet tree to find the expansion in the descendent subspaces. Branches in the tree correspond to indices: scale, and location (frequency), and thus to the sequence of the filters in the quadrature mirror filter pair (QMF). Each row in of coefficients in the tree is computed from the row above it by one application of either of the QMF, which we think of as either summing or differencing operations. The height of each branch is proportional to improvement in cost function obtained by splitting, which means that a tall/long tree branch will show a signficant improvement for that set of coefficients on that row. The entropy drop tree is a graphical display of the route chosen to find the best basis.



The Time-Frequency Phase-Plane Display

The next display is the `Time-Frequency Phase-Plane` plot, which shows the signal sampled in time and frequency. The brighter the *cell* (or more dense, depending on what color table is used in IDL), the more *energy* is in that sample. In addition, the smaller the coverage of colored cells in the time-frequency plane, the better is the compression of the original signal. This display graphically shows how the best wavelet packet found localizes the energy. The next psfile=is such a display for the Sunspots dataset.

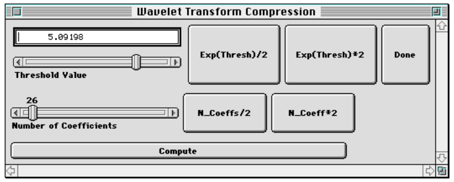


The wavelet packet analysis operation works on 1D datasets, but not currently on 2D datasets.

6. Compression Action

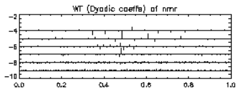
6.1. 1D Data/Signals DWT Compression

This section describes how to compress and reconstruct your 1D data using discrete wavelet transforms.

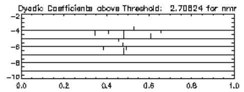


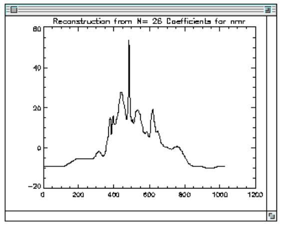
Here you can move the sliders and buttons to change the threshold level and the number of coefficients used in reconstructing the data. Each time you set the sliders or press a button, the values for the threshold and number of coefficients changes. When you are ready to reconstruct your signal, press the **Compute** button.

The plots to pay attention to are the "WT (Dyadic coeffs)" plots and the "Reconstructed" signal or dataset. For example, the transform coefficients of the original NMR dataset looks like the following.



Try increasing the number of coefficients (**N_Coeffs*2**) and notice the signal is reconstructed by more closely approximating the original signal. The threshold is correspondingly changed. Now try changing the threshold. As the threshold increases, the number of coefficients decreases, and the reconstruction is worse. And vice versa.



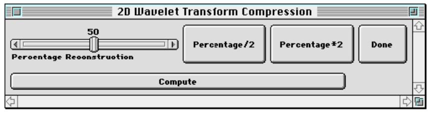


Press the **Done** button on the DWT widget when finished. Note: The data signal/image is not overwritten.

6.2. 2D Data DWT Compression

This section describes how to compress and reconstruct your 2D data using discrete wavelet transforms.

The 2D data operations work in a similar manner to that described for 1D data. Because the compression routines currently are slow, you may want to practice with this 2D function by first reducing the 2D image. Go to **Settings,** and select **Signal Length.** Then choose **64.** This will reduce your image to a 64 by 64 image, which computes much much faster than a 256 by 256 image. Now, select **Wavelet Transform Compression.** A `2D Wavelet Compression` widget will pop up for you to practice reducing the coefficients and reconstructing your image.

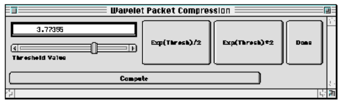


The slider shown gives you the option of choosing a particular reconstruction from a percentage of the coefficients. If you set it to **10%** say, then only **10%** of the coefficients are used in reconstructing your image. If you select **100%,** then all of the coefficients are used.

Press the **Done** button when finished. Note: The data signal/image is not overwritten.

6.3. 1D Data WPT Compression

This section describes how to compress and reconstruct your 1D data using wavelet packet transforms.



This routine works just as the discrete wavelet transform compression described previously. Here you can move the slider and buttons to change the threshold level used in reconstructing the data. Try changing the threshold. As the threshold increases, the number of coefficients decreases, and the reconstruction is worse. And vice versa.

Press the **Done** button when finished with the Packet Compression part. The wavelet packet compression operation works on 1D datasets, but not currently on 2D datasets.

7. DeNoising Action

This action denoises a 1D dataset.

Try selecting the ESCA dataset. Now, normalize it using the **Normalize** action described in the next Section. Finally, push the **DeNoise** button.

Do you see how the general features of the data remain, while the extra wiggles are removed? Now try a different wavelet, say the **Coiflet** with parameter **3.** Push the **DeNoise** button. Does the denoised data now look smoother? Now select a different thresholding technique, say the **MultiHybrid** method. Does this method do a better job? (Actually this method doesn't produce a significantly better job than the **VisuThresh** default method.) Note: this action does not overwrite the dataset.

The denoising operation works on 1D datasets, but not currently on 2D datasets.

8. Miscellaneous Action

These routines perform some simple data manipulations before and/or after the wavelet data manipulations.

8.1. Add Noise

This scaling adds a noise component to the data or signal. The noise is calculated using the parameter set in the **Settings- Noise Type** menu item. If you've chosen **Normal, Uniform, Poisson,** the noise is calculated from a normal, uniform, or poisson distribution, respectively. The default is a **Normal** distribution. The amplitude of the noise is calculated from the **Settings- Noise Level** menu item. Try adding noise to one of the generated signals, then denoise the data. The default is **0.1.** Note: This action *overwrites* the original dataset.

8.2. Normalize

This scaling normalizes the data or signal. In order for the denoising to work properly on naturally-occurring dataset, it must have, at its finest level, a median absolute deviation of 1. Therefore, before trying the denoising action on a real dataset, do this processing first. Note: This action *overwrites* the original dataset.

8.3. DeTrend

This scaling removes any linear trend in a dataset. For a 2D dataset, it detrends row-by-row. Try selecting the ESCA dataset. Then perform this action. Do you see the linear trend removed? Note: This action *overwrites* the original dataset.

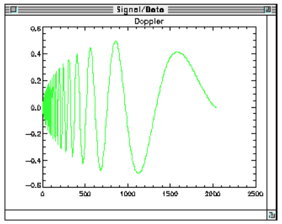
9. Quitting Out of the Program

Press the **Done** button when you are finished with Wavelet Workbench.

1. DeNoising a Signal

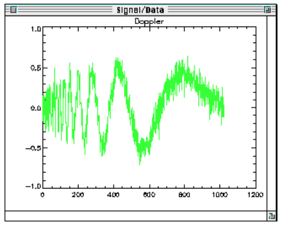
The goal of this experiment is to remove the noise component of a signal. The procedure that we will follow is to select one of the signals: **Doppler** (an attenuating sinusoidal curve with time-varying frequency), add some noise to it, then remove it with the **DeNoise** function. We will demonstrate some differences in denoising due to your choice of a wavelet basis function.

1.) First, select our desired signal. From **Generate 1D Signals,** select **Doppler**. A window titled: `Signal/Data` will appear, displaying the Doppler signal.



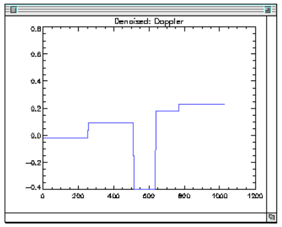
2.) Second, add some noise to the Doppler signal. This part requires us to first select from the **Settings** menu, the noise amplitude) and type (Normal, Uniform, and Poisson. Then we will add noise as a **Misc** action. So to proceed...

From the **Settings** menu, select **Noise Level.** Type **0.1** into the edit box, then press **OK.** Next, from the **Settings** menu, select **Noise Type**. Press **Normal,** then press **OK.** Now, from the **Misc** menu, select **Add Noise**. Do you see your noisy Doppler signal?



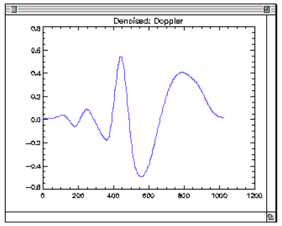
3.) Third, select your desired wavelet. From the **Settings** menu, select **Wavelet**. Choose the **Haar** wavelet, and Parameter **1** when the appropriate widgets appear (don't forget to press **OK** each time!). A window titled: `Wavelet` will appear, displaying the Haar basis function.

4.) Fourth, denoise the signal. We will have to choose a denoising method from the **Settings** menu first. From the **Settings** menu, select **DeNoising Threshold.**. Another widget will appear for you to choose the method **VisuThresh.** Then, press **OK.** Now, press the **DeNoise** button. Several windows will appear displaying the before and after wavelet transform coefficients, and then you'll finally see your denoised signal. Do you see how *blocky* it is?



We want to get rid of the "blockiness" by choosing a smoother wavelet basis function.

So, select a new wavelet. From **Settings** menu, select **Wavelet.** Choose the **Symmlet** wavelet, and Parameter **8** when the appropriate widgets appear. A window titled: `Wavelet` will pop up displaying the Symmlet_8 basis function. Now press the **DeNoise** button. Do you see how much smoother the denoising is?



2. Analyzing Chirp Functions via Wavelet Packets

The goal of this experiment is to use time-frequency analysis to see features in chirp functions that may not be visible using other data analysis techniques. This example was taken from Victor Wickerhauser's Chapter 10.2.3 "Chirps" in his text: Adapted Wavelet Analysis From Theory to Software, (AK Peters, 1994).

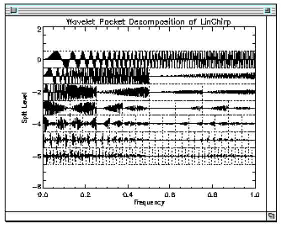
Wavelet Workbench provides several chirp signals. A chirp signal is an oscillatory signal with an increasing or decreasing modulation. The three that we will use in this experiment is a *linear* chirp, a *quadratic* chirp, and *two* chirp. The built-in functions in WWB to generate these signals are given in the following IDL code snippet:

```
t=INDGEN(n)+1
sig_linchirp = SIN(!pi * t * ((n * 0.125) * t))
sig_quadchirp = SIN( (!PI/3.0) * t * (n * t^2))
sig_twochirp = SIN(!pi * t * (n * t)) + SIN((!pi/3.0) * t * (n * t))
```

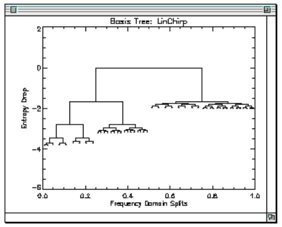
Let's analyze the *linear* chirp signal first.

From the **Generate 1D Signals** menu, select **LinChirp**. A window titled: `Signal/Data` will appear, displaying the linear chirp signal. Now, from the **Time-Scale Actions** menu, select **Wavelet Packet Analysis.** You will see three displays.

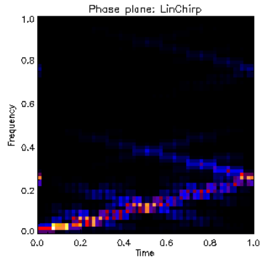
The first display is the `Wavelet Packet Decomposition` display. Essentially, this display shows the original dataset successively scaled and sampled for low and high frequency components. Notice that the second half of every other scale level starting from 1 (labelled "-1" on the y axis) showed an increase in frequency, which is consistent with this sort of signal. See the previous chapter on Wavelet Packet Decomposition for a more complete description on this display.



The second display is the `Entropy Drop Tree` display. Notice that the third-level scaling provided the bigest improvement in finding a wavelet packet basis for this dataset because the lengths of the branches are the tallest.

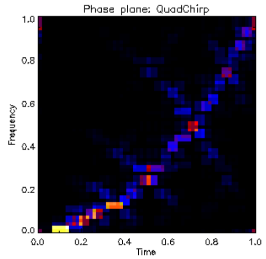


The third display is the one that we're most interested in: the `Phase Plane` display. In the packet analysis leading up this display, Wavelet Workbench chose a basis whose "information cells," i.e. the individually colored pixels, show several lines, indicating the same aspect ratio of time versus frequency. The brightest line is located near the bottom of the display. This sort of linear representation is a classic marker of a linear chirp signal. The other lines above it are "reflected" lines due to the fact that this signal exceeded the Nyquist frequency.



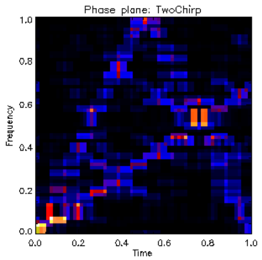
Now let's look at the *quadratic* chirp signal.

From **Generate 1D Signals,** select **QuadChirp**. A window titled: `Signal/Data` will appear displaying the quadratic chirp signal. Now, from the **Time-Scale Actions** menu, select **Wavelet Packet Analysis.** You will see those three displays again. The last `Phase Plane` display will show the most interesting differences. Notice that there is no longer a one-to-one aspect ratio between time and frequency, which gave you the straight lines in the previous linear chirp packet analysis. The relationship between time and frequency look quadratic, as they should be, a marker of this type of quadratic signal. Notice also that we some downward sloping lines, which are due to aliasing of the negative frequencies.



Finally, let's look at the *two* chirp signal.

This signal is the result of superposing *two* linear chirp functions. From the **Generate 1D Signals** menu, select **TwoChirp**. Next, from the **Time-Scale Actions** menu, select **Wavelet Packet Analysis.** You will see similar windows to what you saw before. Look at the last display. Since you have a feel for the output from previous chirp functions, can you tell what is going on here in the `Phase Plane` display? There is one very prominent line starting from the lower left corner, going to the middle of the top of the display window, and then reflected down to the bottom right-hand corner. This reflected line represents aliasing in one of the chirp functions at two times the Nyquist frequency. The second line with a gentler slope represents the second chirp function (whose modulation is milder than the first function) in the superposition of the two chirps. Such a time-frequency analysis as this does well at separating superposed chirp signals.



The descriptions in this chapter show how you can run the Wavelet Workbench library functions from the IDL command line (or calls you can place into your own customized IDL program). So here you will *not* be selecting data and actions from the Wavelet Workbench graphical user-interface (GUI) that is invoked when running `wwb.pro`.

1. WWB Initialization

Many of the routines in the Wavelet Workbench library have COMMON block input and output. Two COMMON blocks should be set before you run any of the routines in the Wavelet Workbench library: `WWB_COMMON` and `Wave_color`. The variable names and ordering in these COMMON blocks are the following.

```
COMMON WWB_COMMON, $
      TEXT_ANNOUNCE, wd, sig, len, st, noise_type, noise_lev, $
      QMF, LD, shrinkage_type, WaveType, ParVal
COMMON Wave_color, $
      max_color, max_image, $
      white, yellow, lavender, aqua, pink, green, red, orange, blue, $
      lt_gray, med_green, brown, olive, purple, dk_gray, black
```

Here is a description of the variables the WWB_COMMON block.

- **sig:** 1-d data/signal array or 2D image
- **QMF:** Quadrature Mirror Filter
- **LD:** Coarseness level (set by WINTWAVE)
- **st:** signal title
- **noise_lev:** noise level.
- **len:** length of data array (2**J size)
- **wd:** working directory
- **noise_type:** noise type (1, 2 or 3 = Normal, Uniform, Poisson, =1 by default)
- **shrinkage_type:** shrinkage method (=1 by default)

```
1 = VisuThresh
2 = MultiHybrid
3 = MultiMAD
4 = MinMaxThresh
5 = WPDeNoise
```

• **WaveType:** Wavelet Type: 'Haar', 'Daubechies', 'Coiflet', 'Symmlet'.
• **ParVal:** Parameter Value of the above wavelet, specific for the wavelet.

```
WaveType = 'Haar', ParVal = 1
WaveType = 'Daubechies', ParVal = 4 or 6
WaveType = 'Coiflet', ParVal = 2 or 3
WaveType = 'Symmlet', ParVal = 6 or 8
```

Some of the global variables, defined in the above COMMON block, should be set. The Wavelet Workbench graphical user-interface program `wwb.pro` sets the global variables to the following values:

```
WaveType = 'Haar'
ParVal = 1
len = 1024
noise_type = 1
noise_lev = .1
shrinkage_type = 1
wd = 'Mac Hard Disk:wwb:'
```

You may wish to use some of these for your own default values. These values can be set by editing the `waveset.pro` program to the values you want. Then run `waveset.pro`.

```
IDL> .run waveset.pro
```

2. Setting Parameters

If you don't wish to use the defaults for the WWB variables, which are set above, you can change them manually at the keyboard. Here, I list variables that you can change via the keyboard.

2.1. Selecting a Wavelet (Quadrature Mirror Filter)

For the specfied wavelet **WaveType**, choose one parameter value **ParVal** as defined below.

```
Haar wavelet:
      IDL> WaveType = 'Haar'
      IDL> ParVal = 1
Daubechies wavelet:
      IDL> WaveType = 'Daubechies'
      IDL> ParVal = 4 or 6
Coiflet wavelet:
      IDL> WaveType = 'Coiflet'
      IDL> ParVal = 2 or 3
Symmlet wavelet:
      IDL> WaveType = 'Symmlet'
      IDL> ParVal = 6 or 8
```

Once you've set a wavelet QMF, run the following procedures:

```
IDL> WDOFILT, WaveType, ParVal, QMF
IDL> WDOWAVE, WaveType,ParVal
```


2.2. Setting the Noise Type

For your desired noise type, set the variable: **noise_type** to be the value 1, 2, or 3. These set the following noise types:

```
1 ===> Normal
2 ===> Uniform
3 ===> Poisson
```

For example:

```
IDL> noise_type=1
```

2.3. Changing the Length/Size of the Dataset or Signal

For your desired array (or image) length, set the variable: **len** to be 64, or 128, or 256, or 512, or 1024, or 2048 (or any other 2**J length, but don't recommend smaller or larger than the above). For example:

```
IDL> len=512
IDL> WDOSGLEN
```

2.4. Setting the Noise Nevel

For your noise level, set the variable: **noise_lev** to be some floating number (approximately = .1 for 1D data; 4.0 and larger for images). For example:

```
IDL> noise_lev = .2
```

2.5. Setting the Shrinkage Type (method of denoising)

For your desired shrinkage (denoising) method, set the variable: **shrinkage_type** to be the value 1, 2, 3, 4, or 5. These set the following shrinkage types:

```
1 ===> VisuThresh
2 ===> MultiHybrid
3 ===> MultiMAD
4 ===> MinMaxThresh
5 ===> WPDeNoise
```

For example:

```
IDL> shrinkage_type=3
```

3. Select the Data or Signal

3.1. 1D Dataset

To read one of the ready-made 1D Datasets in the WWB library, set the variable **st** to be one of the following strings.

```
Caruso:
      IDL> st = 'caruso'

Sunspots:
      IDL> st = 'sunspots'

Seismic:
      IDL> st = 'seismic'

ESCA:
      IDL> st = 'esca'

Ring:
      IDL> st = 'ringsse'

NMR:
      IDL> st = 'nmr'
```

Then run the following routines to read and see your data.

```
IDL> WREADDAT, 1, st, len, wd, sig
IDL> pt = 'Data Set: '+st
IDL> WPLOTIT, pt, 1, sig
```

3.2. 1D Generated

To generate one of the ready-made 1D signals in the WWB library, set the variable **st** to be one of the following strings.

```
HeaviSine:
      IDL> st = 'HeaviSine'

Bumps:
      IDL> st = 'Bumps'

Blocks
      IDL> st = 'Blocks'

Doppler:
      IDL> st = 'Doppler'

Ramp:
      IDL> st = 'Ramp'

Cusp:
      IDL> st = 'Cusp'

Sing:
      IDL> st = 'Sing'
```



```
HiSine:
      IDL> st = 'HiSine'

LoSine:
      IDL> st = 'LoSine'

LinChirp:
      IDL> st = 'LinChirp'

TwoChirp:
      IDL> st = 'TwoChirp'

QuadChirp:
      IDL> st = 'QuadChirp'

MishMash:
      IDL> st = 'MishMash'
```

Then run the following routines to generate and see your data.

```
IDL> sig = WMKSGNAL(st ,len)
IDL> WPLOTIT, st, 1, sig
```

3.3. 2D Dataset

To read one of the ready-made 2D Datasets in the WWB library, set the variable **st** to be one of the following strings.

```
Daubechies:
      IDL> st = 'Daubechies'

Fingerprint:
      IDL> st = 'Fingerprint'

Lincoln:
      IDL> st = 'Lincoln'

MRIScan:
      IDL> st = 'MRIScan'
```

Then run the following routines to read and see your data.

```
IDL> WREADIMG, 1, st, len, wd, sig
IDL> WPLOTIT, st, 0, sig
```

3.4. User-defined 1D Dataset

To read in one of your own sets of 1D data, set the variable **st** to be the name of the data file and doc file (see Chapter 3), and run the `WREADDAT` and `WPLOTIT` procedures as before. E.g.:

```
IDL> st = name  ;string
IDL> WREADDAT, 2, st, len, wd, sig
IDL> pt = 'Data Set: '+st
IDL> WPLOTIT, pt, 1, sig
```

3.5. User-defined 2D Dataset

To read in one of your own sets of 2D data, set the variable **st** to be the name of the data file and doc file (see Chapter 3), and run the WREADIMG and WPLOTIT procedures as before. E.g.:

```
IDL> st = name ;string
IDL> WREADIMG, 2, st, len, wd, sig
IDL> pt = 'Data Set: '+st
IDL> WPLOTIT, pt, 0, sig
```

4. Actions

The chapter covers the wavelet actions that the Wavelet Workbench library provides.

Before you run an "action" on your dataset, you must have data! If you've started scripting your own data-reduction routines, you may want to add the following statements into your IDL program to automatically check that you have data.

```
sz = n_elements(sig)
CASE sz OF
        0: undef = 'True'
        ELSE: undef = 'False'
ENDCASE
IF undef EQ 'False' THEN {do ACTIONS} <== Add your desired actions here
```

4.1. Scalegram

See Chapter 3 for a description of the Scalegram action. To perform it on your dataset while not using the GUI provided, do the following.

```
Truncate sig array appropriately and get QMF
      IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
Call scalegram
      IDL> WDOSCEG
```

4.2. Discrete Wavelet Transform (DWT)

See Chapter 3 for a description of the Discrete Wavelet Transform (DWT) action. To perform it on your dataset, do the following.

```
Truncate sig array appropriately and get QMF
      IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
Run the DWT
      IDL> WDODWT, sig, QMF, LD, st, 3, wc
```


4.3. Scalogram

See Chapter 3 for a description of the Scalogram action. To perform it on your dataset, do the following.

```
Truncate sig array appropriately and get QMF
      IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
Run the Scalogram
      IDL> WDOSCOG
```

4.4. Multi-Resolution Analysis

See Chapter 3 for a description of the Multi-Resolution Analysis action. To perform it on your dataset, do the following.

```
Truncate sig array appropriately and get QMF
      IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
Run the Multi-Resolution Analysis
      IDL> WDOMRA
```

4.5. Wavelet Packet Analysis

See Chapter 3 for a description of the Wavelet Packet Analysis action. To perform it on your dataset, do the following.

```
Truncate sig array appropriately and get QMF
      IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
Run the Wavelet Packet Analysis
      IDL> WDOWPA
```

4.6. Wavelet Transform Compression

See Chapter 3 for a description of the Wavelet Transform Compression action. To perform it on your dataset, do the following.

```
Truncate sig array appropriately and get QMF
      IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
Run the Wavelet Transform Compression
      IDL> WDOWTC
```

4.7. Wavelet Packet Compression

See Chapter 3 for a description of the Wavelet Packet Compression action. To perform it on your dataset, do the following.

```
Truncate sig array appropriately and get QMF
      IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
Run the Wavelet Packet Compression
      IDL> WDOWPC
```

4.8. DeNoise

See Chapter 3 for a description of the DeNoising action. To perform it on your dataset, do the following.

```
Truncate sig array appropriately and get QMF
      IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
(Note: you need to normalize your data first, if working with natural data)
      IDL> sig = WNNOISE(sig,QMF, tt)
      IDL> If tt EQ ‘S’ then WPLOTIT, st, 1, sig
Run the DeNoising
      IDL> WDOSHRNK
```

4.9. Add Noise

See Chapter 3 for a description of the Add Noise action. To perform it on your dataset, do the following.

```
Truncate sig array appropriately and get QMF
      IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
Find out size of dataset
      IDL> ncol = WGETSIZE(sig, nrow)
Set the dataset type (‘S’ = signal, ‘I’ = image)
      IDL> tt = ‘S’ if 1D dataset or tt = ‘I’ if 2D dataset
Run the Add Noise action
      IDL> WADNOISE, sig, nrow, ncol, noise_type, noise_lev, tt
See the result plotted
      IDL> If tt EQ ‘S’ then WPLOTIT, st, 1, sig ELSE WPLOTIT, st, 10, sig
```

4.10. Normalize

See Chapter 3 for a description of the Normalize action. To perform it on your dataset, do the following.

```
Truncate sig array appropriately and get QMF
      IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
Set the dataset type (‘S’ = signal, ‘I’ = image)
      IDL> tt = ‘S’ if 1D dataset or tt = ‘I’ if 2D dataset
Run the Normalize action
      IDL> sig = WNNOISE(sig,QMF, tt)
See the result plotted
      IDL> If tt EQ ‘S’ then WPLOTIT, st, 1, sig ELSE WPLOTIT, st, 10, sig
```

4.11. DeTrend

See Chapter 3 for a description of the DeTrending action. To perform it on your dataset, do the following.


```
Truncate sig array appropriately and get QMF
      IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
Set the dataset type (‘S’ = signal, ‘I’ = image)
      IDL> tt = ‘S’ if 1D dataset or tt = ‘I’ if 2D dataset
Run the DeTrending
      IDL> sig = WDETREND(sig, tt)
See the result plotted
      IDL> If tt EQ ‘S’ then WPLOTIT, st, 1, sig ELSE WPLOTIT, st, 10, sig
```

NAME:
WACONV

PURPOSE:
This function performs a Convolution for Two-Scale Transform.

CATEGORY:
Wavelets

CALLING SEQUENCE:
y = WACONV(f,x)

INPUTS:
f: filter.
x: 1-d signal. i,

OUTPUTS:
y: filtered result.

NOTES:
Filtering by periodic convolution of x with the time-reverse of f.

EXAMPLE:

SEE ALSO:
wiconv, wupdydhi, wupdydlo, wdndydhi, wdndydlo

MODIFICATION HISTORY:
Written by: Amara Graps October, 1994
(Translated from MatLab Wavelab routine: aconv.m)

NAME:
WADNOISE

PURPOSE:
To add noise to the "sig" data array or image.

CATEGORY:
Wavelets

CALLING SEQUENCE:
WADNOISE, sig, numrow, numcol, noise_type, noise_level

INPUTS:
sig = current data array/signal
numrow = number of rows
numcols = number of columns
noise_type = uniform (2) or normal (1) or Poisson (3) (default = 1)
noise_lev = amplitude of noise (default = 1)

OUTPUTS:
sig = now noisy data array/signal/image
TT = 'I' if image and 'S' if signal

EXAMPLE:

```
IDL> ;Truncate sig array appropriately and get QMF
IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
IDL> ncol = WGETSIZE(sig, nrow)
IDL> tt = 'S' if 1D dataset or tt = 'I' if 2D dataset
IDL> WADNOISE, sig, nrow, ncol, noise_type, noise_lev, tt
IDL> If tt EQ 'S' then WPLOTIT, st, 1, sig ELSE WPLOTIT, st, 10, sig
```

MODIFICATION HISTORY:
Amara Graps September 1995.
copyright (c) Amara Graps 1995, 1996.

NAME:
WAVESET

PURPOSE:
This function sets the common variables used in the Wavelet Workbench library.

CATEGORY:
Wavelets.

CALLING SEQUENCE:

```
IDL> .run waveset
```

INPUTS:
none

OUTPUTS:
The common variables set for Wavelet Workbench.

COMMON BLOCKS:

```
COMMON WWB_COMMON, $
TEXT_ANNOUNCE, wd, sig, len, st, noise_type, noise_lev, $
QMF, LD, shrinkage_type, WaveType, ParVal
COMMON Wave_color, $
      max_color, max_image, $
      white, yellow, lavender, aqua, pink, green, red, orange, blue, $
      lt_gray, med_green, brown, olive, purple, dk_gray, black
```

NOTES: If one runs this program first, they can use the Wavelet Workbench routines without the widget interface provided in the routine: wwb.pro

MODIFICATION HISTORY:
Written by: Amara Graps, January 1996.

NAME:
WBBASIS

PURPOSE:
To find the best basis using the Coifman-Wickerhauser Best-Basis Algorithm.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
btree = WBBASIS(tree,D,vtree)

INPUTS:
tree = stat-tree (output by WSTATREE)
D = maximum depth of tree-search

OUTPUTS:
vtree = value of components of best basis
vtree(1) holds value of best basis

EXAMPLE:

SEE ALSO:
wstatree

NOTICE:
Yale University has filed a patent application for this algorithm. Commercial Development based on this algorithm should be cleared by Yale University. Contact them for liscensing information.

MODIFICATION HISTORY:
Written by: Amara Graps December 1994/September1995
(Translated from MatLab Wavelab routine: bestbasis.m)

NAME:
WCALCEPS

PURPOSE:
To compute the distance between 1.0 and the next largest floating point number. The output is the equivalent of Matlab's system "eps" variable.

CATEGORY:
Numerical Precision

CALLING SEQUENCE:
WCALCEPS, eps

INPUTS:
None.

OUTPUTS:
eps = the distance between 1.0 and the next largest single precision floating point number.

NOTES:
This eps value is useful for testing for round-off tolerance. Discussion of this variable is in: Numerical Methods for Physicists, by Alejandro Garcia, Prentice-Hall, 1994.

MODIFICATION HISTORY:
copyright (c) Amara Graps 1995, 1996.

NAME:
WCOMBINE

PURPOSE:
To combine all elements in a matrix, like Matlab's "(:)".

CATEGORY:
Simple math.

CALLING SEQUENCE:
Result = WCOMBINE(A)

INPUTS:
A: A matrix

OUTPUTS:
The matrix turned into a column with all elements combined.

SIDE EFFECTS:
If A is an array, it is left unchanged.

EXAMPLE:

```
IDL>p = [[2,3],[4,5],[6,7]]
IDL>print, p
      2       3
      4       5
      6       7
IDL>print, WCOMBINE(p)
2
4
6
3
5
7
```

MODIFICATION HISTORY:

NAME:
WCUMSUM

PURPOSE:
To sum all of the elements in an array or matrix. If A is a matrix, then sum(A) is a row vector with the sum over each previous element in the column. This function is an equivalent of Matlab's wcumsum function.

CATEGORY:
Simple math.

CALLING SEQUENCE:
Result = WCUMSUM(A)

INPUTS:
A: A vector or matrix

OUTPUTS:
The cumulative sum of the vector or matrix.

EXAMPLE:

```
IDL>a = [[8,2,6],[3,6,7],[4,9,3]]
IDL>print, a
      8 2 6
      3 6 7
      4 9 3
IDL>print, WCUMSUM(a)
     8.000     2.000     6.000
    11.000     8.000    13.000
    15.000    17.000    16.000
```

EXAMPLE:

```
IDL>a = [1, 2, 3, 4 5, 6, 7, 8, 9, 10]
IDL>print, WCUMSUM(a)
     1    3    6   10   15   21   28   36   45   55
```

MODIFICATION HISTORY:
Amara Graps, September 1995

NAME:
WDETREND

PURPOSE:
This function removes a linear bias from the "sig" data array.

CATEGORY:
Wavelets

CALLING SEQUENCE:
out = WDETREND(sig)

INPUTS:
sig = current data array/signal

OUTPUTS:
out = now detrended data array/signal
TT = 'I' if image and 'S' if signal

EXAMPLE:

```
IDL> ;Truncate sig array appropriately and get QMF
IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
IDL> tt = 'S' if 1D dataset or tt = 'I' if 2D dataset
IDL> ;Detrend dataset
IDL> sig = WDETREND(sig, tt)
IDL> See the detrended data
IDL> If tt EQ 'S' then WPLOTIT, st, 1, sig ELSE WPLOTIT, st, 10, sig
```

MODIFICATION HISTORY:
Amara Graps September 1995.

NAME:
WDNDYDHI

PURPOSE:
This function performs a Hi-Pass Downsampling operation (periodized).

CATEGORY:
Wavelets

CALLING SEQUENCE:
d = WDNDYDHI(x,f)

INPUTS:
x: 1-d signal at fine scale.
f: filter

OUTPUTS:
y: 1-d signal at coarse scale.

EXAMPLE:

SEE ALSO:
wdndydlo, wupdydhi, wupdydlo, wfwtpo, wiconv

MODIFICATION HISTORY:
Written by: Amara Graps October, 1994
(Translated from MatLab Wavelab routine: downdyadhi.m)

NAME:
WDNDYDLO

PURPOSE:
This function performs a Lo-Pass Downsampling operation. (periodized).

CATEGORY:
Wavelets

CALLING SEQUENCE:
d = WDNDYDLO(x,f)

INPUTS:
x: 1-d signal at fine scale.
f: filter

OUTPUTS:
y: 1-d signal at coarse scale.

EXAMPLE:

SEE ALSO:
wdndydhi, wupdydhi, wupdydlo, wfwtpo, waconv

MODIFICATION HISTORY:
Written by: Amara Graps October, 1994
(Translated from MatLab Wavelab routine: downdyadlo.m)

NAME:
WDODWT

PURPOSE:
This procedure performs a discrete wavelet transform and plots the coefficients.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
WDODWT, X_work, QMF, LD, plotwin, wc

INPUTS:
X_work: 1-d signal array
QMF: Quadrature Mirror Filter
LD: lowest resolution level
st: signal title
plotwin: plot window to plot into

OUTPUTS:
wc = the array of wavelet coefficients
Plus the discrete wavelet transform coeffs plotted to the screen.

SEE ALSO:
WINTWAVE (necessary for this procedure to get QMF)

EXAMPLE:

```
Truncate sig array appropriately and get QMF
IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
Calculate Discrete Wavelet Transform
IDL> WDODWT, sig, QMF, LD, st, 3, wc
```

MODIFICATION HISTORY:
Written by: Amara Graps August/September, 1995

NAME:
WDOFILT

PURPOSE:
This procedure defines the quadrature mirror filter used for all subsequent computations.

CATEGORY:
Wavelets

CALLING SEQUENCE:
WDOFILT, WaveType, ParVal, QMF

INPUTS:
WaveType: String of choices: Haar, Daubechies, Coiflet, Symmlet
ParVal: Parameter value for a specific wavelet

OUTPUTS:
QMF: Quadrature Mirror Filter

MODIFICATION HISTORY:
Written by: Amara Graps August, 1995

NAME:
WDOMRA

PURPOSE:
This procedure performs a multi-resolution depiction of the data or signal.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
WDOMRA

INPUTS: (via COMMON)
sig: 1-d signal array
QMF: Quadrature Mirror Filter
st: signal title
LD: multiresolution scale

OUTPUTS:
The discrete wavelet transform coeffs plotted to the screen.

SEE ALSO:
WINTWAVE (necessary for this procedure to get QMF)

EXAMPLE:

```
IDL> ;Truncate sig array appropriately and get QMF
IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
IDL> ;Call Multi-resolution analysis
IDL> WDOMRA
```

COMMON BLOCKS:
COMMON WWB_COMMON, TEXT_ANNOUNCE, wd, sig, len, st, noise_type, noise_lev, QMF, LD, shrinkage_type, WaveType, ParVal

EXAMPLE:

MODIFICATION HISTORY:
Written by: Amara Graps August/September, 1995

NAME:
WDOSCEG

PURPOSE:
This procedure generates a scalegram and plots it. The scalegram is the wavelet analog of the power spectrum. It is just the (squared) wavelet coefficient averaged over all location index values at a fixed value of the scale index. The sums of squares results in a flat scalegram if input = white noise

CATEGORY:
Wavelets.

CALLING SEQUENCE:
WDOSCEG

INPUTS: (via COMMON)
X_work: 1-d signal array
QMF: Quadrature Mirror Filter

OUTPUTS:
The scalegram plotted to the screen.

COMMON BLOCKS:
Wave_color, and 16 colors, i.e.:
COMMON Wave_color, max_color, max_image, white, yellow, lavender, aqua, pink, green, red, orange, blue, lt_gray, med_green, brown, olive, purple, dk_gray, black
COMMON WWB_COMMON, TEXT_ANNOUNCE, wd, sig, len, st, noise_type, noise_lev, QMF, LD, shrinkage_type, WaveType, ParVal

SEE ALSO:
init_scalegram (necessary for this procedure)

EXAMPLE:

```
Truncate sig array appropriately and get QMF
IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
Call scalegram
IDL> WDOSCEG
```

NOTES:
We prefer to use the Haar wavelet because:
1. it is the only one that you can use down to the "finest scale" ... i.e., width of wavelet = 2 X sampling interval.
2. No worries about end-effects (i.e. no wavelet sticking out over the ends of the data!)

3. The correction of the resulting scalegram has a very simple form (= mean counting rate) for Poisson statistics in the data.

For noisy data, the discontinuity and nondifferentiability of the Haar wavelet don't seem to be important deficiencies.

MODIFICATION HISTORY:
Written by: Amara Graps August/September, 1995
(Translated from MatLab Wavelab routine: do_scalegram.m)

NAME:
WDOSCOG

PURPOSE:
This procedure generates a scalogram and plots it. The scalogram is a moderately well-established term for (a plot usually, of) the wavelet coefficients as a function of the two independent variables, or indices in the discrete case, scale and location. It is common to use the absolute value, or even the log of the absolute value in practice.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
WDOSCOG

INPUTS: (via COMMON)
X_work: 1-d signal array
QMF: Quadrature Mirror Filter

OUTPUTS:
The scalegram plotted to the screen.

COMMON BLOCKS:
Wave_color, and 16 colors, i.e.:
COMMON Wave_color, max_color, max_image, white, yellow, lavender, aqua, pink, green, red, orange, blue, lt_gray, med_green, brown, olive, purple, dk_gray, black
COMMON WWB_COMMON, TEXT_ANNOUNCE, wd, sig, len, st, noise_type, noise_lev, QMF, LD, shrinkage_type, WaveType, ParVal

SEE ALSO:
WINTWAVE (necessary for this procedure)

EXAMPLE:

```
Truncate sig array appropriately and get QMF
IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
Call scalogram
IDL> WDOSCOG
```

MODIFICATION HISTORY:
Written by: Amara Graps August, 1995
(Translated from MatLab Wavelab routine: do_scalogram.m)

NAME:
WDOSGLEN

PURPOSE:
To resize the "sig" data array (if it exists), or set it for future calculations.

CATEGORY:
Wavelets

CALLING SEQUENCE:
WDOSGLEN

INPUTS: (via COMMON)
sig = current data array/signal or image resized (if it exists)

OUTPUTS: (via COMMON)
out = now wdetrended data array/signal
TT = 'I' if image and 'S' if signal

COMMON BLOCKS:
COMMON WWB_COMMON, TEXT_ANNOUNCE, wd, sig, len, st, noise_type, noise_lev, QMF, LD, shrinkage_type, WaveType, ParVal

MODIFICATION HISTORY:
Amara Graps September 1995.

NAME:
WDOSHRNK

PURPOSE:
This procedure denoises a time series, using wavelet shrinkage, and plots the result. The time series is the working dataset, the wavelet is the current one selected, and the shrinkage method is the one chosen under the Wavelet Workbench Settings: DeNoising Threshold.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
WDOSHRNK

NECESSARY INPUTS (via COMMON) :
sig: 1-d signal array
QMF: Quadrature Mirror Filter
LD: Coarseness level (set by WINTWAVE)
st: signal title
noise_lev: noise level (set by NoLev procedure, =0 by default)
noise_type: noise type (set by NoType procedure,
1, 2 or 3 = Normal, Uniform, Poisson, =1 by default)
shrinkage_type: shrinkage method (set by Threshold procedure, =1 by default)

```
1 = VisuThresh
3 = MultiHybrid
4 = MultiMAD
5 = MinMaxThresh
6 = WPDeNoise
```

WaveType: Wavelet Type: 'Haar', 'Daubechies', 'Coiflet', 'Symmlet'

OUTPUTS:
The denoised time series is plotted to the screen.

COMMON BLOCKS:
COMMON WWB_COMMON, TEXT_ANNOUNCE, wd, sig, len, st, noise_type, noise_lev, QMF, LD, shrinkage_type, WaveType, ParVal

SEE ALSO:
WINTWAVE (necessary for this procedure)

EXAMPLE:

```
IDL> ;Truncate sig array appropriately and get QMF
IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
IDL> ;Need to normalize first, if working with natural data)
IDL> sig = WNNOISE(sig,QMF, tt)
IDL> If tt EQ 'S' then WPLOTIT, st, 1, sig
IDL> WDOSHRNK
```

MODIFICATION HISTORY:
Written by: Amara Graps September-November 1995
(Translated from MatLab Wavelab routine: do_shrinkage.m)

NAME:
WDOWAVE

PURPOSE:
This procedure calls the MakeWavelet procedure and plots the result.

CATEGORY:
Wavelets

CALLING SEQUENCE:
WDOWAVE, WaveType, ParVal

INPUTS:
WaveType: Type of Wavelet: 'Haar' 'Daubechies' 'Coiflet' 'Symmlet'
ParVal: Parameter value corresponding to the wavelet

OUTPUTS:
A simple wavelet plot

MODIFICATION HISTORY:
Written by: Amara Graps August, 1995

NAME:
WDOWPA

PURPOSE:
This procedure performs a wave packet analysis depiction of the data or signal.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
WDOWPA

INPUTS: (via COMMON)
sig:1-d signal array
QMF: Quadrature Mirror Filter
st: signal title

OUTPUTS:
A wavelet packet decomposition, an entropy drop tree, and a phase plane all plotted to the screen in different windows.

COMMON BLOCKS:
COMMON WWB_COMMON, TEXT_ANNOUNCE, wd, sig, len, st, noise_type, noise_lev, QMF, LD, shrinkage_type, WaveType, ParVal

SEE ALSO:
WINTWAVE (necessary for this procedure to get QMF)

EXAMPLE:

```
IDL> ;Truncate sig array appropriately and get QMF
IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
IDL> ;Call Wavelet Packet Analysis
IDL> WDOWPA
```

MODIFICATION HISTORY:
Written by: Amara Graps August/Sepember, 1995

NAME:
WDOWPC

PURPOSE:
This procedure performs a wave packet compression of the data or signal.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
WDOWPC

INPUTS: (via COMMON)
sig: 1-d signal array
QMF: Quadrature Mirror Filter
st: signal title

OUTPUTS:
A wavelet packet decomposition, an entropy drop tree, and a phase plane all plotted to the screen in different windows, along with a widget for you to control compression parameters, and the final compressed representation of your data.

COMMON BLOCKS:
COMMON WWB_COMMON, TEXT_ANNOUNCE, wd, sig, len, st, noise_type, noise_lev, QMF, LD, shrinkage_type, WaveType, ParVal

SEE ALSO:
WINTWAVE (necessary for this procedure to get QMF)

EXAMPLE:

```
IDL> ;Truncate sig array appropriately and get QMF
IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
IDL> ;Call Wavelet Packet Compression
IDL> WDOWPC
```

MODIFICATION HISTORY:
Written by: Amara Graps Sepember, 1995

NAME:
WDOWTC

PURPOSE:
This procedure sets up runs a simple event loop for the Wavelet Transform Compression action.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
WDOWTC

INPUTS: (via COMMON)
sig: 1-d signal array
QMF: Quadrature Mirror Filter
st: signal title

OUTPUTS:
A wavelet transform, a widget for you to control compression parameters, and the final compressed representation of your data.

COMMON BLOCKS:
COMMON WWB_COMMON, TEXT_ANNOUNCE, wd, sig, len, st, noise_type, noise_lev, QMF, LD, shrinkage_type, WaveType, ParVal
COMMON WTC_common, The block for the two sliders (FSLID3, FSLID4) the four buttons (BUTTON4, BUTTON5, , BUTTON8, BUTTON9) and array of wavelet coeffs (wc), sorted array of wavelet coeffs (wcwmsort), threshold value (thr), the max number of coeffs (n), the current value of the sliders (threshslider, numslider), and the coarsest level for the wavelet transform (Coarse)

EXAMPLE:

```
IDL> ;Truncate sig array appropriately and get QMF
IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
IDL> ;Do wavelet transform compression (puts up widget)
IDL> WDOWTC
```

MODIFICATION HISTORY:
Written by: Amara Graps, August/September 1995.

NAME:
WDYAD

PURPOSE:
This function indexes entire j-th dyad of 1-d wavelet xform.

CATEGORY:
Wavelets

CALLING SEQUENCE:
i = WDYAD(j)

INPUTS:
j: integer

OUTPUTS:
i: integer list (array) of all indices of wavelet coeffts at j-th level

EXAMPLE:

MODIFICATION HISTORY:
Written by: Amara Graps October, 1994
(Translated from MatLab Wavelab routine: dyad.m)

NAME:
WDYAD2IX

PURPOSE:
This function converts wavelet indexing into linear indexing..

CATEGORY:
Wavelets

CALLING SEQUENCE:
ix = WDYAD2IX(j,k)

INPUTS:
j: Resolution Level. j ge 0.
k: Spatial Position. 0 le k lt 2**j.

OUTPUTS:
ix = index in linear 1-d wavelet transform array where the (j,k) wavelet coefficient is stored.

EXAMPLE:

MODIFICATION HISTORY:
Written by: Amara Graps October, 1994
(Translated from MatLab Wavelab routine: dyad2ix.m)

NAME:
WDYADLNG

PURPOSE:
This function finds length and dyadic length of an array.

CATEGORY:
Wavelets

CALLING SEQUENCE:
n = WDYADLNG(x,J)

INPUTS:
x: array of length n = 2**J (hopefully)

OUTPUTS:
n: length(x)
J: least power of two greater than n

SIDE EFFECTS:
A warning is issued if n is not a power of 2.

EXAMPLE:

SEE ALSO:
wquadlen, wdyad, wdyad2ix

MODIFICATION HISTORY:
Written by: Amara Graps October, 1994
(Translated from MatLab Wavelab routine: dyadlength.m)

NAME:
WFPTWP

PURPOSE:
This function analyzes dataset into specific wavelet packet basis.

CATEGORY:
Wavelets

CALLING SEQUENCE:
coef = WFPTWP(basis,x,qmf)

INPUTS:
basis = tree selecting wavelet packet basis
x = 1-d image to be transformed into basis
qmf = quadrature mirror filter

OUTPUTS:
coef =1-d wavelet packet coeffts in given basis

EXAMPLE:

MODIFICATION HISTORY:
Written by: Amara Graps December, 1994
(Translated from MatLab Wavelab routine: fpt_wp.m)

NAME:
WFWT2PO

PURPOSE:
This function performs a 2-dimensional wavelet transform (periodized, orthonormal).

CATEGORY:
Wavelets

CALLING SEQUENCE:
wc =WFWT2PO(x,L,qmf)

INPUTS:
x: 2-d image (n by n array, n dyadic)
L: coarse level
qmf: quadrature mirror filter

OUTPUTS:
wc: 2-d wavelet transform coefficients

NOTES:
A two-dimensional Wavelet Transform is computed for the array x. To reconstruct, use WIWT2PO.

EXAMPLE:

SEE ALSO:
wiwt2po, wmkofilt

MODIFICATION HISTORY:
Written by: Amara Graps October, 1994
(Translated from MatLab Wavelab routine: fwt2_po.m)

NAME:
WFWTPO

PURPOSE:
This function generates a Forward Wavelet Transform (periodized, orthonormal).

CATEGORY:
Wavelets

CALLING SEQUENCE:
wc = WFWTPO(x,L,qmf)

INPUTS:
x: 1-d signal; length(x) = 2**J
L: Coarsest level scale L much less than J
qmf: quadrature mirror filter (orthonormal), (qmf defines wavelet shape

OUTPUTS:
wc: 1-d wavelet transform of x. (Gives wavelet coefficients)

NOTES:
1. QMF filter may be obtained from WMKOFILT
2. usually, length(qmf) lt 2**(L+1)
3. To reconstruct use WIWTPO

EXAMPLE:

SEE ALSO:
wiwtpo, wmkofilt

NOTES: (on j, k)
The location k, at level j, runs from 0 to 2**j-1. A signal of length n is taken to a set of n wavelet coefficients by the (orthogonal) wavelet transformation. Since the wavelet coefficients are indexed in a dyadic way due to the natural location/scale interpretation inherent with wavelets, you cannot pick a (j,k) combination that doesn't exist for the transformed data. Consequently j ranges as above and k can be at most log_2(n). The wavelet transformation is not over-complete (i.e., it takes a signal of length n to a set of coefficients of length n).

MODIFICATION HISTORY:
Written by: Amara Graps October, 1994
(Translated from MatLab Wavelab routine: fwt_po.m)

NAME:
WGETSIZE

PURPOSE:
To determine from a matrix or array, the ncol and nrow..

CATEGORY:
Simple array typing.

CALLING SEQUENCE:
ncol = WGETSIZE(Array nrow)

INPUTS:
Array: array from we wish to find ncol and nrow.

OUTPUTS:
nrow: number of rows
ncol: number of columns

EXAMPLE:

```
IDL> a=[[1,2,3],[4,5,6]]
IDL> print, a
       1       2       3
       4       5       6
IDL> ncol=WGETSIZE(a, nrow)
IDL> print, ncol
           3
IDL> print, nrow
           2
IDL> b=[1,2,3,4,5,6,7]
IDL> print, b
       1       2       3       4       5       6       7
IDL> ncol=WGETSIZE(b, nrow)
IDL> print, ncol
           7
IDL> print, nrow
           1
```

MODIFICATION HISTORY:
Amara Graps September 1995.
(c) copyright Amara Graps 1995, 1996.

NAME:
WGETUSER

PURPOSE:
This procedure goes to the two folders within the WWB folder that may or may not contain user-defined data and returns two string arrays that contain the titles of the files. The two folders that this function checks are: 'u1ddata' and 'u2ddata'. If file(s) are in 'u1ddata'/'u2ddata', then the user1d/user2D string array returns strings that precede the '.doc' filenames.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
WGETUSER, wd, user1d, user2d, nf1, nf2

INPUTS:
wd = working directory (string)

OUTPUTS:
user1d = string array containing user-defined titles of the 1d files.
user2d = string array containing user-defined titles of the 2d files.
nf1 = number of 1D user-defined files
nf2 = number of 2D user-defined files

EXAMPLE:

MODIFICATION HISTORY:
Written by: Amara Graps September, 1995

NAME:
WHARDTHR

PURPOSE:
This function applies a hard threshold.

CATEGORY:
Wavelets

CALLING SEQUENCE:
x = WHARDTHR(y,t)

INPUTS:
y = Noisy Data
t = Threshold

OUTPUTS:
x = abs(y) gt t

SEE ALSO:
WHARDTHR, WSOFTTHR, WDOSHRNK

EXAMPLE:

MODIFICATION HISTORY:
Written by: Amara Graps December, 1994
(Translated from MatLab Wavelab routine: hardthresh.m)

NAME:
WHYBRTHR

PURPOSE:
To perform Shrinkage to Wavelet Coefficients via Adaptive Threshold Selection Using Principle of Stein's Unbiased Risk Estimate Variant without Rigorous Proof

CATEGORY:
Wavelets.

CALLING SEQUENCE:
xhat = WHYBRTHR(Y)

INPUTS:
y = Noisy Data with Std. Deviation = 1

OUTPUTS:
xhat = Estimate of mean vector

EXAMPLE:

NOTES:
See Donoho, D.L. and Johnstone, I.M. "Adapting to Unknown Smoothness by Wavelet Shrinkage"

SEE ALSO:
WMHYBRID, WDOSHRNK

MODIFICATION HISTORY:
Written by: Amara Graps December, 1994
(Translated from MatLab Wavelab routine: hybridthresh.m)

NAME:
WICONV

PURPOSE:
This function performs a Convolution for Two-Scale Transform.

CATEGORY:
Wavelets

CALLING SEQUENCE:
y = WICONV(f,x)

INPUTS:
f: filter. x: 1-d signal.

OUTPUTS:
y: filtered result.

NOTES:
Filtering is a periodic convolution of x with f.

EXAMPLE:

SEE ALSO:
waconv, wupdydhi, wupdydlo, wdndydhi, wdndydlo

MODIFICATION HISTORY:
Written by: Amara Graps October, 1994
(Translated from MatLab Wavelab routine: iconv.m)

NAME:
WIMAGE

PURPOSE:
This procedure displays an image with provisions for
1. plot title, 2. annotated x and y axis, 3. simplified OPLOT capability.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
WIMAGE, a, {, {title = title}, {xtitle = xtitle}, {ytitle = ytitle},
{xrange = xrange}, {yrange = yrange}, {noframe = noframe}, {aspect} }

INPUTS:
a = image quantity

OPTIONAL INPUTS:
title = plot title
xtitle = x axis title
ytitle = y axis title
xrange = array spanning entire x axis range.
NOTE: WIMAGE only uses min(XRANGE) and max(XRANGE).
yrange = array spanning entire y axis range.
NOTE: WIMAGE only uses min(YRANGE) and max(YRANGE).
aspect = the x over y aspect ratio of the output image if not set aspect is set to (size_x/size_y) of the input image.
noframe = if set do not draw axis box around image

OUTPUTS:
An image plot with colored rectangles based image quantity.

EXAMPLE:

NOTES:
WIMAGE first determins the size of the plot data window with a dummy call to PLOT. When the output device is "X", CONGRID is used to scale the image to the size of the available data window. Otherwise, if the output device is Postscript, scaleable pixels are used in the call to TV. PLOT draws the x and y axis and titles.

MODIFICATION HISTORY:
Paul Ricchiazzi oct92
Earth Space Research Group, UCSB
Amara Graps, January 1995
Bay Area Environmental Research Institute/NASA-Ames
(Adapted for WaveLab 0.6)

NAME:
WINTWAVE

PURPOSE:
This procedure initializes important wavelet calculation parameters.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
WINTWAVE, x_work, WType, Par, QMF, LD

INPUTS:
X_work: 1-d signal array WType: Wavelet Type (string = 'Haar', 'Daubechies', 'Coiflet', 'Symmlet')
Par: Wavelet parameter depends on WType.

```
Options are:
'Haar'-->Par = 2
'Daubechies' -->Par = 4, 6
'Coiflet' -->Par = 2, 3
'Symmlet' -->Par = 6, 8
```

OUTPUTS:
X_Work: truncated at 2**n length
QMF: Quadrature Mirror Filter
LD: Lowest resolution level

SEE ALSO:
wdosceg (uses the output from this procedure)

MODIFICATION HISTORY:
Written by: Amara Graps August/September, 1995

NAME:
WIPTWP

PURPOSE:
This function synthesizes a signal from wavelet packet coefficients.

CATEGORY:
Wavelets

CALLING SEQUENCE:
x = WIPTWP(btree,coef,qmf)

INPUTS:
basis = basis tree selecting wavelet packet basis
coef = coefficients in that basis
qmf = quadrature mirror filter

OUTPUTS:
x: 1-d signal reconstructed from coefficients

NOTES:
1. qmf filter may be obtained from WMKOFILT
2. usually, length(qmf) lt 2̂(L+1)

EXAMPLE:

SEE ALSO:
wfptwp, wmkofilt

MODIFICATION HISTORY:
Written by: Amara Graps February, 1995
(Translated from MatLab Wavelab routine: ipt_wp.m)

NAME:
WIWT2PO

PURPOSE:
This function performs an inverse 2-dimensional wavelet transform (periodized, orthonormal).

CATEGORY:
Wavelets

CALLING SEQUENCE:
x = WIWT2PO(wc,L,qmf)

INPUTS:
wc: 2-d wavelet transform [n by n array, n dyadic]
L: coarse level
qmf: quadrature mirror filter

OUTPUTS:
x: 2-d signal reconstructed from wc.

NOTES:
If wc is the result of a forward 2d wavelet transform, with wc = FWT2_PO(x,L,qmf) then x = WIWT2PO(wc,L,qmf) reconstructs *exactly*, if qmf is a nice quadrature mirror filter, e.g. one made by WMKOFILT.

EXAMPLE:

SEE ALSO:
fwt2_po, wmkofilt

MODIFICATION HISTORY:
Written by: Amara Graps November, 1994
(Translated from MatLab Wavelab routine: iwt2_po.m)

NAME:
WIWTPO

PURPOSE:
This function generates an Inverse Wavelet Transform (periodized, orthonormal).

CATEGORY:
Wavelets

CALLING SEQUENCE:
x = WIWTPO(wc,L,qmf).

INPUTS:
wc: 1-d wavelet transform: length(wc)= 2**J.
L: Coarsest Level; L much less than J
qmf: quadrature mirror filter (orthonormal)

OUTPUTS:
x: 1-d signal reconstructed from wc.

NOTES:
1. qmf filter may be obtained from WMKOFILT
2. usually, length(qmf) lt 2**(L+1)

EXAMPLE:

SEE ALSO:
wfwtpo, wmkofilt

MODIFICATION HISTORY:
Written by: Amara Graps November, 1994
(Translated from MatLab Wavelab routine: iwt_po.m)

NAME:
WLCKAXES

PURPOSE:
This procedure creates an axis and sets the xmin,xmaxymin,ymax

CATEGORY:
Plotting.

CALLING SEQUENCE:
WLCKAXES, a

INPUTS:
a: 4-element array containing x and y endpoints for the axes

OUTPUTS:
An axis

MODIFICATION HISTORY:
Written by: Amara Graps August, 1995
(Translated from MatLab Wavelab routine: LockAxes.m)

NAME:
WLNSPACE

PURPOSE:
This function creates an array of N equally spaced points between x1 and x2.

CATEGORY:
Wavelets

CALLING SEQUENCE:
y = WLNSPACE(x1, x2, n)

INPUTS:
x1: value of 1st element.
n: number of points
x2: value of last element.

OUTPUTS:
y: An array of of uniformly-spaced values.

SEE ALSO:
wmkiarry, wmkfarry

MODIFICATION HISTORY:
Written by: Amara Graps BAER, December, 1994

NAME:
WLSHIFT

PURPOSE:
This function performs a circular left shift of 1-d signal.

CATEGORY:
Wavelets

CALLING SEQUENCE:
l = WLSHIFT(x)

INPUTS:
x: 1-d signal

OUTPUTS:
l: left-shifted 1-d signal

EXAMPLE:

PROCEDURE:
l(i-1) = x(i) except l(n-1) = x(0)

MODIFICATION HISTORY:
Written by: Amara Graps October, 1994
(Translated from MatLab Wavelab routine: lshift.m)

NAME:
WMEAN

PURPOSE:
This function computes the mean of any array

CATEGORY:
G1- Simple calculations on statistical daa.

CALLING SEQUENCE:
Result = WMEAN(Array)

INPUTS:
Array: The data array. Array may be any type except string.

OUTPUTS:
MEAN returns the mean of the values in the data array.

EXAMPLE:

```
IDL> a = [3, 9, 2.3, 10, 15, 3]
IDL> mm = WMEAN(a)
IDL>print, mm
      7.05000
```

PROCEDURE:
Mean = TOTAL(Array)/N_ELEMENTS(Array)

MODIFICATION HISTORY:
Amara Graps, BAER, Sept. 1994.

NAME:
WMFILT

PURPOSE:
This function performs a convolution operation that runs a window through a data set. The result is equivalent to Matlab's call: filter(filt, 1, x).

CATEGORY:
Wavelets

CALLING SEQUENCE:
y = WMFILT(filt, x)

INPUTS:
x: 1-d signal.
filter: filter/window

OUTPUTS:
y: 1-d signal smeared with the filter/window function.

EXAMPLE:

SEE ALSO:
wiconv, waconv

MODIFICATION HISTORY:
Written by Amara Graps with help from Jeff Scargle, NASA-Ames. October, 1994

NAME:
WMHYBRID

PURPOSE:
To perform Shrinkage to Wavelet Coefficients via Adaptive Threshold Selection Using Principle of Stein's Unbiased Risk Estimate Variant without Rigorous Proof

CATEGORY:
Wavelets.

CALLING SEQUENCE:
x = WMHYBRID(wc,L)

INPUTS:
wc = Wavelet Transform coefficients of noisy sequence with N(0,1) noise
L = low-frequency cutoff for Wavelet Transform

OUTPUTS:
ws = result of applying WHYBRTHR to each Dyadic Block

EXAMPLE:

SEE ALSO:
whybrthr, wdoshrnk

MODIFICATION HISTORY:
Written by: Amara Graps December, 1994
(Translated from MatLab Wavelab routine: multihybrid.m)

NAME:
WMIRFILT

PURPOSE:
This function applies a (-1)**t modulation.

CATEGORY:
Wavelets

CALLING SEQUENCE:
h = WMIRFILT(l)

INPUTS:
x: 1-d signal

OUTPUTS:
h: a 1-d signal with DC frequency content shifted to the Nyquist frequency.

EXAMPLE:

SEE ALSO:
wdndydhi

PROCEDURE:
h(t) = (-1)**(t-1) * x(t), 1 le t le length(x)

MODIFICATION HISTORY:
Written by: Amara Graps October, 1994
(Translated from MatLab Wavelab routine: mirrorfilt.m)

NAME:
WMKFARRY

PURPOSE:
This function creates a FRACTIONAL array with the same notation that Matlab uses in creating one, i.e., y = a:b:c.

CATEGORY:
Wavelets

CALLING SEQUENCE:
y = WMKFARRY(a, b, c)

INPUTS:
a: value of 1st element.
b: increment (usually 1).
c: value of last element.

OUTPUTS:
y: A fractional array.

SEE ALSO:
wmkiarry (array of integer index values)

EXAMPLE:

```
IDL> t=WMKFARRY(.5,.5,2.5)
IDL> print, t
       .5       1.0       1.5       2.0       2.5
```

MODIFICATION HISTORY:
Written by: Amara Graps August, 1995

NAME:
WMKIARRY

PURPOSE:
This function creates an INTEGER INDEX array with the same notation that Matlab uses in creating one, i.e. y = a:b:c.

CATEGORY:
Wavelets

CALLING SEQUENCE:
y = WMKIARRY(a, b, c)

INPUTS:
a: index of 1st element.
b: increment (usually 1).
c: index of last element.

OUTPUTS:
y: An array of index values.

NOTES:
Matlab indices for arrays begin with 1, IDL begins with 0. This routine assumes that the array created has (index-1) of what Matlab uses. I.e. If Matlab assumes an index array of: [2,6,10,14,18], this routine creates: [1,5,9,13,17].

SEE ALSO:
wmkfarry (arrays of fractional values)

EXAMPLE:

```
IDL> t=WMKIARRY(1,1,5)
IDL> print, t
       0       1       2       3       4
```

MODIFICATION HISTORY:
Written by: Amara Graps November, 1994

NAME:
WMKOFILT

PURPOSE:
This function generates an orthonormal QMF filter for a wavelet transform.

CATEGORY:
Wavelets

CALLING SEQUENCE:
qmf = WMKOFILT(Type, Par)

INPUTS:
Type: string, 'Haar', 'Daubechies', 'Coiflet', 'Symmlet'
Par: integer, e.g. if Type ='Coiflet', Par=3 specifies Coiflet3

OUTPUTS:
qmf: quadrature mirror filter

EXAMPLE:

SEE ALSO:
wfwtpo, wiwtpo, fwt2_po, iwt2po, wpanalys

MODIFICATION HISTORY:
Written by: Amara Graps October, 1994
(Translated from MatLab Wavelab routine: makeonfilter.m)

NAME:
WMKSGNAL

PURPOSE:
This function generates an artificial signal.

CATEGORY:
Wavelets

CALLING SEQUENCE:
sig = WMKSGNAL(Name, n)

INPUTS:
Name: string = ‘HeaviSine’, ‘Bumps’, ‘Blocks’ ‘Doppler’, ‘Ramp’, ‘Cusp’, ‘Sing’, ‘HiSine’, ‘LoSine’, ‘LinChirp’, ‘TwoChirp’, ‘QuadChirp’ ‘MishMash’
n: desired signal length

OUTPUTS:
sig: 1-d signal

EXAMPLE:

REFERENCES:
various articles of D.L. Donoho and I.M. Johnstone

MODIFICATION HISTORY:
Written by: Amara Graps August, 1995
(Translated from MatLab Wavelab routine: makesignal.m)

NAME:
WMKWVLET

PURPOSE:
This function generates a periodized orthogonal wavelet.

CATEGORY:
Wavelets

CALLING SEQUENCE:
wave = WMKWVLET(j,k [,Family=Famly,par=par,Gender=Gender,n=n])

INPUTS:
j, k: location-scale indices: 2**-j approx width of wavelet k/2**j approx location of wavelet on unit interval

OUTPUTS:
wave: 1-d signal, the wavelet

KEYWORD PARAMETERS:
FAMILY: string, ‘Haar’, ‘Daubechies’, ‘Coiflet’, ‘Symmlet’
PAR: integer parameter to WMKOFILT, e.g. if Family =’Coiflet’, Par=3 specifies Coiflet3
GENDER: ’Mother’,’Father’
N: signal length (dyadic)

EXAMPLE:

MODIFICATION HISTORY:
Written by: Amara Graps November, 1994
(Translated from MatLab Wavelab routine: makewavelet.m)

NAME:
WMMIN

PURPOSE:
To find the minimum elements of an array or matrix. This function works the same way as Matlab's MIN: [y,ind]=MIN(array) works.

CATEGORY:
Math.

CALLING SEQUENCE:
Result = WMMIN(array, ind)

INPUTS:
Array: The data array. Array may be any type except string.

OUTPUTS:
Result = WMMIN returns the lowest value of an array, or if the array is a matrix, it returns the lowest value of each *column* in the matrix.
ind = the index of the lowest value(s).

NOTES:
IDL's indices are one less than Matlab's.

EXAMPLE:

```
IDL>array=[[1,3,2],[6,3,4],[9,1,0]]
IDL>print, array
       1       3       2
       6       3       4
       9       1       0
IDL>y= WMMIN(array, ind)
IDL>print, y
1     1     0
IDL>print, ind
0     2     2
```

MODIFICATION HISTORY:
Amara Graps, BAER, December 1994.

NAME:
WMSORT

PURPOSE:
To create a increasing-order sorted array. This function works the same way as Matlab's SORT.

CATEGORY:
Math.

CALLING SEQUENCE:
Result = WMSORT(Array)

INPUTS:
Array: The data array. Array may be any type except string.

OUTPUTS:
WMSORT returns an increasing-order sorted array from the input data array.

PROCEDURE:
WMSORT = REVERSE(SORT(SORT(array)))

EXAMPLE:

```
IDL>a=[[1,3,2],[6,3,4],[9,1,0]]
IDL>print, a
       1       3       2
       6       3       4
       9       1       0
IDL>print, WMSORT(a)
1     1     0
6     3     2
9     3     4
```

MODIFICATION HISTORY:
Amara Graps, BAER, December 1994.

NAME:
WMTHRESH

PURPOSE:
To Perform Minimax Thresholding

CATEGORY:
Wavelets.

CALLING SEQUENCE:
x = WMTHRESH (y)

INPUTS:
y =signal upon which to perform thresholding

OUTPUTS:
x =result

EXAMPLE:

SEE ALSO:
wdoshrnk

NOTES:
See article “Ideal Spatial Adaptation via Wavelet Shrinkage” By D.L. Donoho and I.M. Johnstone

MODIFICATION HISTORY:
Written by: Amara Graps December, 1994
(Translated from MatLab Wavelab routine: minmaxthresh.m)

NAME:
WMULTMAD

PURPOSE:
To perform Shrinkage to Wavelet Coefficients via level-dependent Noise level estimation.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
ws = WMULTMAD(wc,L)

INPUTS:
wc = Wavelet Transform coeffcients of noisy sequence
L = low-resolution cutoff for Wavelet Transform

OUTPUTS:
ws = result of applying WVISUTHR to each wavelet level, after scaling so MAD of coefficienst at each level = .6745

EXAMPLE:

SEE ALSO:
wvisuthr, wdoshrnk

MODIFICATION HISTORY:
Written by: Amara Graps December, 1994
(Translated from MatLab Wavelab routine: multimad.m)

NAME:
WNNOISE

PURPOSE:
This function normalizes the signal to noise level 1.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
y = WNNOISE(x,QMF)

INPUTS:
x = 1-d/2-d signal
QMF = quadrature mirror filter

OUTPUTS:
y = 1-d signal, scaled so wavelet coefficients at finest level have median absolute deviation 1
TT = 'I' if image and 'S' if signal

EXAMPLE:

```
IDL> ;Truncate sig array appropriately and get QMF
IDL> WINTWAVE, sig, WaveType, ParVal, QMF, LD
IDL> tt = 'S' if 1D dataset or tt = 'I' if 2D dataset
IDL> sig = WNNOISE(sig,QMF, tt)
IDL> If tt EQ 'S' then WPLOTIT, st, 1, sig ELSE WPLOTIT, st, 10, sig
```

NOTES:
This is required pre-processing to use any of the DeNoising tools on naturally-occurring data.

SEE ALSO:
wpdnoise

MODIFICATION HISTORY:
Written by: Amara Graps September, 1995
(Translated from MatLab Wavelab routine: normnoise.m)

NAME:
WNNORM

PURPOSE:
To compute the Euclidean norm of an column-dominant or a row-dominant array.

CATEGORY:
Simple math.

CALLING SEQUENCE:
Result = WNNORM(A)

INPUTS:
A: A vector

OUTPUTS:
The Euclidean norm of the vector.

NOTES:
To get the same result as IDL's intrinsic NORM function, one can run REFORM on the column-dominant array. This function eliminates that step.

EXAMPLE:

```
IDL>a = [[1,2,3],[4,5,6],[7,8,9]]
IDL>print, a
     1       2       3
     4       5       6
     7       8       9
IDL>b = a(0,*)
IDL>print, b
       1
       4
       7
IDL>print, WNNORM(b)
       8.12404
```

MODIFICATION HISTORY:
Amara Graps, January 1995.

NAME:
WNODE

PURPOSE:
To perform tree indexing.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
index = WNODE(d,b)

INPUTS:
d = depth from root of tree
b = index among the 2$\hat{d}$ possibilities in a left-right scan at that depth

OUTPUTS:
index = linear index of node in tree structure

EXAMPLE:

SEE ALSO:
wbbasis, wstatree

MODIFICATION HISTORY:
Written by: Amara Graps December 1994/August1995
(Translated from MatLab Wavelab routine: node.m)

NAME:
WPACKET

PURPOSE:
This function performs packet table indexing.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
p = WPACKET(d,b,n)

INPUTS:
d = depth of splitting in packet decomposition
b = block index among 2**d possibilities at depth d
n = length of signal

OUTPUTS:
p = linear indices of all coeff's in that block

MODIFICATION HISTORY:
Written by: Amara Graps August, 1995
(Translated from MatLab Wavelab routine: packet.m)

NAME:
WPANALYS

PURPOSE:
This function creates a dyadic table of all Wavelet Packet coefficients.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
wp = WPANALYS(x,D,qmf)

INPUTS:
x = array of dyadic length n=2Ĵ
D = degree of finest frequency partition
QMF = orthonormal quadrature mirror filter

OUTPUTS:
wp = array of Wavelet Packet Decompositions
Coefficients for frequency interval [b/2*d,(b+1)/2*d] is stored in wp(WPACKET(d,b,n),d+1)

SEE ALSO:
wintwave (necessary for this procedure to get QMF), wpsynth, wpdnoise, wdndydhi, wdndydlo

REFERENCE:

```
title   = {Signal processing and compression with wavelet packets},
author  = {R.R. Coifman and Y. Meyer and S. Quake and M.V. Wickerhauser},
booktitle = {Progress in wavelet analysis and applications,
              Proceedings of the International Conference ‘‘Wavelets and
              Applications", Toulouse, 1992},
editor = {Y. Meyer and S. Roques},
publisher = {Editions Frontieres},
year    = 1993,
pages   = {77-93}
```

MODIFICATION HISTORY:
Written by: Amara Graps August, 1995
Translated from MatLab Wavelab routine: wpanalysis.m

NAME:
WPAROUND

PURPOSE:
To perform a circular map onto 0 to (n-1)

CATEGORY:
Array Transformation.

CALLING SEQUENCE:
indnew = WPAROUND(ind, n)

INPUTS:
ind = vector to wrap
n = length of wrap

OUTPUTS:
indnew = An array with the values wrapped.

EXAMPLE:

```
IDL>arr = [1,2,3,4,5,6,7,8,9]
IDL>PRINT, arr
       1       2       3       4       5       6       7
       8       9
IDL>n = 7
IDL>indnew = WPAROUND(arr,n)
IDL>PRINT, indnew
       1       2       3       4       5       6       7
       1       2
```

MODIFICATION HISTORY:
Written by: Amara Graps November, 1995
(Translated from MatLab Wavelab routine: wraparound.m)

NAME:
WPDNOISE

PURPOSE:
This function denoises in an adaptively chosen WP basis

CATEGORY:
Wavelets.

CALLING SEQUENCE:
clean = WPDNOISE(x,D,QMF,bb,st)

INPUTS:
x = 1-d signal to be de-noised.
D = maximum depth of basis tree
QMF = quadrature mirror filter for freq. splitting

OUTPUTS:
clean = cleaned signal
bb = basistree naming basis in which de-noising was done
st = stat tree: statistics driving basis search

NOTES:
1. Assumes noise level eq 1
2. Uses Stein Unbiased Estimate of risk to evaluate basis
3. Uses Coifman-Wickerhauser BestBasis algorithm to select best basis

MODIFICATION HISTORY:
Written by: Amara Graps December, 1994
(Translated from MatLab Wavelab routine: wpdenoise.m)

NAME:
WPHPLANE

PURPOSE:
This procedure partitions phase space by rectangular blocks.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
WPHPLANE, TFtype,btree,pkt,titlestr=titlestr,nTFR=nTFR,qmf=qmf

INPUTS:
TFtype = string: type of TF-packets used ('WP','CP')
btree = basis tree
pkt = wavelet or cosine packet table
titlestr = signal name (optional)
nTFR = number of x-points in phaseplane image (optional)
qmf = qmf for calculating WP phase plane location (optional)

OUTPUTS:
An image plot with colored rectangles based on recursive dyadic partitioning of y axis according to splits in basis tree .

EXAMPLE:
To create a "Heisenberg image" plot of data analyzed with WPANALYS, make the following call:

```
IDL>WPHPLANE,'WP',btree,wp,titlestr = st, nTFR = 64
```

where 'WP' is wavelet packet, btree and wp are outputs from WPANALYS, st is the string identifier for the data ('seismic', 'sunspots', etc.), nTFR is the number of boxes (dyads!) on a side. Keep nTFR on the low side, If nTFR = 256, or higher you'll be waiting 5 minutes or more for output.

NOTE:
This routine also performs a Cosine-Packet Analysis ('CP'), however, cosine packet analysis isn't currently a function fully implemented in Wavelet Workbench. So consider this 'CP' option a 'hook' for now.

MODIFICATION HISTORY:
Written by: Amara Graps November, 1995
(Translated from MatLab Wavelab routine: imagephaseplane.m)

NAME:
WPIPULSE

PURPOSE:
This function calculates location of wavelet packet entry.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
dwp = WPIPULSE(wp,d,b,k,qmf)

INPUTS:
wp = packet table, ignored except for size
d,b,k = 1-d packet-index
qmf = quadrature mirror filter

OUTPUTS:
dwp = packet table, the complete wavelet packet analysis of wavelet packet (d,b,k)

MODIFICATION HISTORY:
Written by: Amara Graps November, 1995
(Translated from MatLab Wavelab routine: wpimpulse.m)

NAME:
WPKBCOEF

PURPOSE:
This function stuffs basis coefficients into packet table

CATEGORY:
Wavelets.

CALLING SEQUENCE:
pkt = WPKBCOEF(basis,opkt,x)

INPUTS:
basis = basis tree selecting basis the coefficients come from
opkt = wavelet or cosine packet table
x = 1-d signal, coefficients in the basis of btree

OUTPUTS:
pkt = new packet table with coefficients in specified basis filled in from x

MODIFICATION HISTORY:
Written by: Amara Graps September, 1995
(Translated from MatLab Wavelab routine: packbasiscoeff.m)

NAME:
WPLBTREE

PURPOSE:
This procedure displays basis tree with decorated branch lengths

CATEGORY:
Wavelets.

CALLING SEQUENCE:
WPLBTREE, btree,Depth,stree,st

INPUTS:
basis = basis tree (e.g. output of WBBASIS)
Depth = max depth of tree
stree = stat tree of pkt table
st = signal title

OUTPUTS:
A plot of tree with height of each branch proportional to improvement in cost function obtained by splitting

MODIFICATION HISTORY:
Written by: Amara Graps November, 1995
(Translated from MatLab Wavelab routine: plotbasistree.m)

NAME:
WPLMLRES

PURPOSE:
This procedure generates a Multi-Resolution Display of a 1-d wavelet transform. This multi-resolution depiction of the signal is the type referenced in S. Mallat's work.

CATEGORY:
Wavelets

CALLING SEQUENCE:
WPLMLRES, wc, L, scal, qmf, st

INPUTS:
wc: 1-d wavelet transform
L: level of coarsest scale
scal: scale factor [0 == autoscal]
qmf: quadrature mirror filter used to make wc
st: title of signal/dataset

OUTPUTS:
a Multi-Resolution Display of 1-d Wavelet Transform

SEE ALSO:
wplwcoef, wfwtpo, wiwtpo, wmkofilt

MODIFICATION HISTORY:
Written by: Amara Graps November, 1994
(Translated from MatLab Wavelab routine: plotmultires.m)

NAME:
WPLMSURF

PURPOSE:
This procedure generates a Multi-Resolution surface display of a 1-d wavelet transform.

CATEGORY:
Wavelets

CALLING SEQUENCE:
WPLMSURIF,wc,LD,scal,qmf, st

INPUTS:
wc: 1-d wavelet transform
LD: level of coarsest scale
scal: scale factor [0 == autoscal]
qmf: quadrature mirror filter used to make wc
st: title of signal/dataset

OUTPUTS:
a Multi-Resolution Surface Display of 1-d Wavelet Transform

SEE ALSO:
wplmlres, wfwtpo, wiwtpo, wmkofilt

MODIFICATION HISTORY:
Written by: Amara Graps September/November, 1995

NAME:
WPLOTIT

PURPOSE:
This procedure plots or images (using TVSCL) a 1D array or a 2D image.

CATEGORY:
Wavelets

CALLING SEQUENCE:
WPLOTIT, pt, win, data

INPUTS:
pt = a plot title string
win = window number to plot into
=0 for Image,
=1 for Signal/Data,
=2 for Wavelet,
=10 or 11 for stretched image,
other number = just a window and no plotting
data = data array or image to be plotted

OUTPUTS:
plotted data array in window number

MODIFICATION HISTORY:
Written by: Amara Graps September 1995
copyright (c) Amara Graps 1995, 1996.

NAME:
WPLPKTBL

PURPOSE:
This procedure display entries in wavelet and cosine packet tables.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
WPLPKTBL, pkt,scal,signaltitle

INPUTS:
pkt = wavelet or cosine packet table
scal = optional scaling factor (0 =gt autoscale)
st = title of signal or dataset

OUTPUTS:
spike plot of coefficients in packet table

SEE ALSO:
wintwave (necessary for this procedure to get QMF) wpanalys, wpsynth, wptour, wdndydhi, wdndydlo

MODIFICATION HISTORY:
Written by: Amara Graps August, 1995
(Translated from MatLab Wavelab routine: plotpackettable.m)

NAME:
WPLSPIKE

PURPOSE:
This procedure plots a 1-d signal as baseline with series of spikes.

CATEGORY:
Wavelets

CALLING SEQUENCE:
WPLSPIKE,base,t,x, signaltitle , First

INPUTS:
base: number, baseline level
t: ordinate values
x: 1-d signal, specifies spike deflections from baseline
First: First time through (will erase data) = 'True' otherwise
= 'False' (will use oplot, and keep previous plotted data).
signaltitle: title for signal

OUTPUTS:
a 1-d signal plot as baseline with series of spikes.

SEE ALSO:
wplwcoef, wplpktbl

MODIFICATION HISTORY:
Written by: Amara Graps August, 1995
(Translated from MatLab Wavelab routine: plotspikes.m)

NAME:
WPLWCOEF

PURPOSE:
This procedure plots a spike-plot display of wavelet coefficients.

CATEGORY:
Wavelets

CALLING SEQUENCE:
WPLWCOEF, wc, L, scal, signaltitle

INPUTS:
wc: 1-d wavelet transform
L: level of coarsest scale
scal: scale factor [0 = autoscale]
signaltitle: title for plot

OUTPUTS:
A display of wavelet coefficients (coarsest level NOT included) by level and position.

SEE ALSO:
wplmlres, wfwtpo, wiwtpo, wplspike

NOTE:
L controls the lowest resolution level. The lowest resolution level will be L out of log_2(n). For example, if L=0, the lowest resolution level will have 2**L+1 elements in it, which will represent the overall average. We often use L=3.

MODIFICATION HISTORY:
Written by: Amara Graps November, 1994
(Translated from MatLab Wavelab routine: plotwavecoeff.m)

NAME:
WPOISIMG

PURPOSE:
This procedure adds Poisson noise to an array.

CATEGORY:
Image simulation.

CALLING SEQUENCE:
RESULT = WPOISIMG (IMAGE[, SEED])

INPUT:
IMAGE: A numeric array (byte, integer, long, or float) of arbitrary dimensionality. This is the array of values around which values in the result will be Poisson-distributed.

OPTIONAL INPUTS:
SEED: A longword seed for the random number generator. If this is not supplied, the value -123456789L is used for generating the first random value.

KEYWORD PARAMETER:
OUTPUT_KIND: The data type of the output array, that is byte, integer, longword, or float. The words “byte”, “int”, “integer”,“long”, “longword”, and “float”, in upper or lower case, are accepted, as are the numeric IDL values 1,2,3,4 for byte, integer, longword, and float.

OUTPUT:
WPOISIMG returns a copy of the input array, Poisson noise added.

RESTRICTIONS:
Negative input values are mapped to a result of 0. A call is made to the routine WSTBUSY to provide messages once a minute during the processing of large arrays. You can eliminate that line of code if you want to.

PROCEDURE:
Create an image with values Poisson-distributed around the mean values IMAGE, using Knuth’s “Algorithm Q”. (Donald E. Knuth, The Art of Computer Prograwmming, Volume 2, “Seminumerical Algorithms”, Addison- Wesley (1969), page 117. This routine IS NOT VECTORIZED, AND SHOULD RUN SLOWLY. A deft IDL’er could probably vectorize the algorithm, and anyone who does so is entitled to a gold star. Where the gaussian and Poisson distributions are essentially identical (mean value gt 50) a normal, that is gaussian, distribution is used.

EXAMPLE:
Here is how you can create a 100x100 array of values Poisson-distributed around the mean value 5.0, and check the empirical probability against the Poisson distribution:

```
n = 100
a = REPLICATE(5.0,n,n)
b = WPOISIMG( a, out="byte" )
TVSCL, b
print, STDEV(b,mean), mean, SQRT(5.0)
h = HISTOGRAM( b )
prob = FLOAT(h)/N_ELEMENTS(b)
probi = EXP(-5.0)
for i=0,10 do begin PRINT, i, PROB(i), probi & $
probi=probi*5.0/(i+1)
```

MODIFICATION HISTORY:

Written by: James Hamill
Siemens Medical Systems
2501 N. Barrington Rd.
Hoffman Estates, IL 60195-7372
(708)304-7760
hamill@sgi.siemens.com
February, 1992
Renamed from poisson_image.pro by A. Graps to go with Wavelet Workbench, March 1996.

NAME:
WPSYNTH

PURPOSE:
This function synthesize a signal from wavelet packet coefficients.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
x = WPSYNTH(basis,wp,qmf)

INPUTS:
basis = basis tree indicating wavelet packet to use
wp = wavelet packet table
qmf = quadrature mirror filter

OUTPUTS:
x = 1-d signal whose wavelet packet coeff's in basis btree are in wp

SEE ALSO:
wintwave (necessary for this procedure to get QMF) wpanalys, wpdnoise, wdndydhi, wdndydlo

REFERENCE:

```
title   = {Signal processing and compression with wavelet packets},
author  = {R.R. Coifman and Y. Meyer and S. Quake and M.V. Wickerhauser},
booktitle = {Progress in wavelet analysis and applications,
              Proceedings of the International Conference ``Wavelets and
              Applications", Toulouse, 1992},
editor = {Y. Meyer and S. Roques},
publisher = {Editions Frontieres},
year    = 1993,
pages   = {77-93}
```

MODIFICATION HISTORY:

Written by: Amara Graps September, 1995
(Translated from MatLab Wavelab routine: wpsynthesis.m)

NAME:
WQUAD2IX

PURPOSE:
This function translates 2-d wavelet indexing into linear indexing.

CATEGORY:
Wavelets

CALLING SEQUENCE:
kx = WQUAD2IX(j, k1, k2, ep, ky)

INPUTS:
j: resolution level
k1, k2: spatial indices
ep: spatial orientation

```
ep = 0:  Father(x)Father(y)
     1:  Mother(x)Father(y)
     2:  Father(x)Mother(y)
     3:  Mother(x)Mother(y)
```

OUTPUTS:
kx,ky = index in 2-d wavelet transform array of indicated wavelet.

SEE ALSO:
fwt2_po, wiwt2po

MODIFICATION HISTORY:
Written by: Amara Graps November, 1994
(Translated from MatLab Wavelab routine: quad2ix.m)

NAME:
WQUADLEN

PURPOSE:
This function finds length and dyadic length of square matrix.

CATEGORY:
Wavelets

CALLING SEQUENCE:
n = WQUADLEN(x,J)

INPUTS:
x: 2-d image; size(n,n), n = 2**J (hopefully)

OUTPUTS:
n: n_elements(x) of one of the sides of the image
J: least power of two greater than n

SIDE EFFECTS:
A warning is issued if n is not a power of 2, or if x is not a square matrix

EXAMPLE:

SEE ALSO:
wdyadlng,wquad2ix

MODIFICATION HISTORY:
Written by: Amara Graps November 1994/September 1995
(Translated from MatLab Wavelab routine: quadlength.m)

NAME:
WRAND

PURPOSE:
This function generates a matrix with ncol, nrow with unformly distributed numbers. This function mimics the Matlab x=rand(size(A)) where size(A) gives you a 2-element array with values (nrow,ncol).

CATEGORY:
Simple math.

CALLING SEQUENCE:
rmatrix = WRAND(seed, nrow, ncol)

INPUTS:
seed: Named variable containing the seed value for the random number generation. If it is undefined, it is derived from the system clock.
nrow: number of rows
ncol: number of columns

OUTPUTS:
A matrix w/ncol, nrow, uniformly distributed.

NOTE:

```
The easiest way to get the input rows/columns is to:
IDL> t=size(A)
IDL> numcol = t(1) ;number of columns
IDL> numrow=t(2) ;number of rows

Or if A is an array:

IDL> numcol = t(1) ;number of columns
IDL> numrow=t(0) ;number of rows(=1)
```

EXAMPLE:

```
IDL> A = [[8,2,6],[3,6,7]]
IDL>print, A
      8 2 6
      3 6 7
IDL> t=SIZE(A)
IDL> numcol = t(1) ;number of columns
IDL> numrow=t(2) ;number of rows
IDL> rmatrix = WRAND(seed, nrow,ncol)
IDL> print, rmatrix
    0.2190    0.6789    0.9347
    0.0470    0.6793    0.3835
```


MODIFICATION HISTORY:
Amara Graps September 1995.

NAME:
WRANDN

PURPOSE:
To generate a matrix with ncol, nrow with normally distributed numbers. This function mimics the Matlab x=rand(size(A)) where size(A) gives you a 2-element array with values (nrow,ncol).

CATEGORY:
Simple math.

CALLING SEQUENCE:
rmatrix = WRANDN(seed, nrow, ncol)

INPUTS:
seed: Named variable containing the seed value for the random number generation. If it is undefined, it is derived from the system clock.
nrow: number of rows
ncol: number of columns

OUTPUTS:
A matrix w/ncol, nrow, normally distributed.

NOTE:

```
The easiest way to get the input rows/columns is to:
IDL> t=SIZE(A)
IDL> numcol = t(1) ;number of columns
IDL> numrow=t(2) ;number of rows

Or if A is an array:

IDL> numcol = t(1) ;number of columns
IDL> numrow=t(0) ;number of rows(=1)
```

EXAMPLE:

```
IDL> A = [[8,2,6],[3,6,7]]
IDL>print, A
      8 2 6
      3 6 7
IDL> t=SIZE(A)
IDL> numcol = t(1) ;number of columns
IDL> numrow=t(2) ;number of rows
IDL> rmatrix = WRANDN(seed, nrow,ncol)
IDL> print, rmatrix
    -0.323532      1.70055    -0.425996
    -0.584817     -1.50207      1.04847
```


MODIFICATION HISTORY:
Amara Graps September 1995.

NAME:
WREADDAT

PURPOSE:
This procedure reads in two files- both ascii- that hold a one-d dataset and the description of that dataset.

CATEGORY:
File Input/Output

CALLING SEQUENCE:
WREADDAT, dtype, st, len, wd, data

INPUTS:
st = a string that directly relates to the two files we will open:
st+'.asc' = the data file
st+'.doc' = the documentation file
wd = the working directory
len = the length of the data array
dtype = 1(given) or 2(user-defined)

OUTPUTS:
len = the length of the data image
Prints out the contents of the st+'.doc' file to the screen.
data = data array to be plotted

EXAMPLE:

```
To read a user defined OneD dataset (2) with 'st' as the file
identifier (e.g. st='phone'), and working directory wd (string)
Type:

IDL> WREADDAT, 1, st, len, wd, sig
```

MODIFICATION HISTORY:
Amara Graps September 1995.

NAME:
WREADIMG

PURPOSE:
This procedure reads in two files- both ascii- that hold a two-d dataset and the description of that dataset.

CATEGORY:
File Input/Output

CALLING SEQUENCE:
WREADIMG, dtype, st, len, wd, image

INPUTS:
st = a string that directly relates to the two files we will open:
st+'.raw' = the data file
st+'.doc' = the documentation file
wd = the working directory
len = the square size of the image
dtype = 1(given) or 2(user-defined)

OUTPUTS:
len = the length of the data image
Prints out the contents of the st+'.doc' file to the screen.
image = image array to be plotted

EXAMPLE:

```
To read a user defined Image dataset (2) with 'st' as the file
identifier (e.g. st='phone'), and working directory wd (string)
Type

IDL> WREADIMG, 2, st, len, wd, image
```

MODIFICATION HISTORY:
Amara Graps September 1995.

NAME:
WREM

PURPOSE:
This function computes the remainder after division.

CATEGORY:
Simple math.

CALLING SEQUENCE:
Result = WREM(x,y)

INPUTS:
x: numerator
y: denominator

OUTPUTS:
The remainder after x is divded by y.

SIDE EFFECTS:
If x is an array, this function operates element by element.

PROCEDURE:
Remainder = x-n*y where n = FIX(x/y) is the integer part of the quotient x/y.

MODIFICATION HISTORY:
Amara Graps, BAER, October 1994.
(c) copyright Amara Graps 1995, 1996.

NAME:
WRESHAPE

PURPOSE:
To wreshape all elements in a matrix to the desired shape, like Matlab's "wreshape".

CATEGORY:
Simple math.

CALLING SEQUENCE:
Result = WRESHAPE(A, nrows,ncols)

INPUTS:
A: A matrix
m: number of rows
n: number of columns

OUTPUTS:
The matrix turned into a column with the elements in the right shape.

SIDE EFFECTS:
If A does not have m*n elements, an error occurs.

EXAMPLE:

```
IDL>a = [[2,3],[4,5],[6,7]]
IDL>print, a
      2       3
      4       5
      6       7
IDL>print, WRESHAPE(a, 2, 3)
      2.00000      6.00000      5.00000
      4.00000      3.00000      7.00000
```

MODIFICATION HISTORY:
copyright (c) Amara Graps 1995, 1996.

NAME:
WRSHIFT

PURPOSE:
This function performs a circular right shift of 1-d signal.

CATEGORY:
Wavelets

CALLING SEQUENCE:
r = WRSHIFT(x)

INPUTS:
x: 1-d signal

OUTPUTS:
r: right-shifted 1-d signal

EXAMPLE:

PROCEDURE:
r(i-1) = x(i-2) except r(0) = x(n-1)

SEE ALSO:
wlshift

MODIFICATION HISTORY:
Written by: Amara Graps November, 1994
(Translated from MatLab Wavelab routine: rshift.m)

NAME:
WSHASROW

PURPOSE:
This function makes a single or multi-dimensional signal a row vector.

CATEGORY:
Wavelets

CALLING SEQUENCE:
row = WSHASROW(sig)

INPUTS:
sig: a row, column vector or matrix

OUTPUTS:
row: a row vector

EXAMPLE:

MODIFICATION HISTORY:
Written by: Amara Graps October, 1994
(Translated from MatLab Wavelab routine: shapeasrow.m)

NAME:
WSHLIKE

PURPOSE:
This function makes a 1-d signal with given shape.

CATEGORY:
Wavelets

CALLING SEQUENCE:
vec = WSHLIKE(sig,proto)

INPUTS:
sig: a row, column vector, or matrix
proto: a prototype shape (row, column vector, or matrix)

OUTPUTS:
vec: a vector with contents taken from sig and same shape as proto

EXAMPLE:

SEE ALSO:
wshasrow

MODIFICATION HISTORY:
Written by: Amara Graps October, 1994
(Translated from MatLab Wavelab routine: shapelike.m)

NAME:
WSIGN

PURPOSE:
To compute the sign of a number. This function mimics Matlab's sign function.

CALLING SEQUENCE:
Result = WSIGN(num)

INPUTS:
Num: Any number or array

OUTPUTS:
WSIGN returns either 1, 0, or -1.

MODIFICATION HISTORY:
Amara Graps, BAER, December 1994.
(c) copyright Amara Graps 1995, 1996.

NAME:
WSIGTYPE

PURPOSE:
This function returns a character identifying the sig array as either a 1D signal "S" (size (n,1)) or matrix "I" (nrow,ncol) where nrow = ncol .

CATEGORY:
Simple math.

CALLING SEQUENCE:
TT = WSIGTYPE(sig, len, error)

INPUTS:
sig: Either a 1D signal or a 2D matrix

OUTPUTS:
A character TT = "I" (Image) or "S" (Signal).
len = length of one side of square image (2D) or length of signal (1D)
error = 1 (OK, square matrix or array) or 0 (not square matrix)

EXAMPLE:

```
IDL> a = [[3,3,3],[3,3,3],[3,3,3]]
IDL> PRINT, a
       3       3       3
       3       3       3
       3       3       3
IDL> b = [1,2,3,4,5,6,7,8,9]
IDL> PRINT, b
       1       2       3       4       5       6       7
       8       9
IDL> TT = WSIGTYPE(a, len, error)
IDL> PRINT, tt, len, error
I           3   1
IDL> TT = WSIGTYPE(b, len, error)
IDL> PRINT, tt, len, error
S           9       1
```

MODIFICATION HISTORY:
Amara Graps September 1995.

NAME:
WSMIN

PURPOSE:
To find the minimum elements of an array or matrix. This function works the same way as Matlab's min function: y=min(array, val) works.

CATEGORY:
Math.

CALLING SEQUENCE:
Result = WSMIN(array, val)

INPUTS:
Array: The data array. Array may be any type except string. val: a value that is compared with every element of Array

OUTPUTS:
Result: WSMIN returns the lowest value of an array element or "val", or if the array is a matrix, it returns the lowest value of each *column* in the matrix or "val".

NOTES:
IDL's indices are one less than Matlab's.

EXAMPLE:

```
IDL>array=[[1,2,3],[4,5,6],[7,8,9]]
IDL>a = 1.4823
IDL>PRINT, array
       1       2       3
       4       5       6
       7       8       9
IDL>y= WSMIN(array, a)
IDL>PRINT, y

    1.0000    1.4823    1.4823
    1.4823    1.4823    1.4823
    1.4823    1.4823    1.4823
```

MODIFICATION HISTORY:
Amara Graps, September 1995.

NAME:
WSOFTTHR

PURPOSE:
This function applies a soft threshold.

CATEGORY:
Wavelets

CALLING SEQUENCE:
x = WSOFTTHR(y,t)

INPUTS:
y = Noisy Data
t = Threshold

OUTPUTS:
x = WSIGN(y) * [((ABS(y) - t) + ABS((ABS(y) - t)))/2]

SEE ALSO:
whardthr, wdoshrnk

MODIFICATION HISTORY:
Written by: Amara Graps December, 1994
(Translated from MatLab Wavelab routine: softthresh.m)

NAME:
WSPECTRM

PURPOSE:
This function generates a power spectrum.

CATEGORY:
Wavelets

CALLING SEQUENCE:
sp = WSPECTRM(x,n)

INPUTS:
x: a 1-d array n: desired length of 1-d array

OUTPUTS:
sp: a power spectrum of input x array

EXAMPLE:

PROCEDURE:
Uses the IDL FFT function.

MODIFICATION HISTORY:
Written by: Amara Graps October, 1994
(Translated from MatLab Wavelab routine: tw_spectrum.m)

NAME:
WSTATREE

PURPOSE:
To build tree with entropy numbers.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
stree = WSTATREE(pkt,Entropy,par)

INPUTS:
pkt = packet table [wavelet or cosine packet] (n x L)
Entropy = type of entropy to use: options are

```
 'Entropy' -- Coifman-Wickerhauser
 'l**p'    -- sum th_i**p, 0 lt p lt 2, p = par
 'N(eps)'  -- # ge eps, eps = par
 'Risk'    -- sum min(th_i**2,eps**2), eps=par
 'SURE'    -- SURE(Thresholding), thresh = par
(Stein's unbiased estimate of risk)
```

par = extra parameter for 'L**P', 'N(EPS)', 'SURE', Entropy calculation

OUTPUTS:
stree = stat tree of entropy numbers.
tree(wnode(d,b)) contains entropy of wpacket(d,b,n)

EXAMPLE:

SEE ALSO:
wbbasis, wplbtree

MODIFICATION HISTORY:
Written by: Amara Graps December 1994/August1995
(Translated from MatLab Wavelab routine: calcstattree.m)

NAME:
WSTBUSY

PURPOSE:
This function can be embedded inside an IDL code that is expected to run for a long time, announcing with printed messages that the code is still running. Unlike a PRINT statement in a loop, WSTBUSY makes its announcements just once a minute.

CATEGORY:
Simple I/O

CALLING SEQUENCE:
WSTBUSY[, ANNOUNCEMENT]

INPUTS:
A: A vector or matrix

OPTIONAL INPUT:
ANNOUNCEMENT: An array of messages or values to be printed. If no parameter is used, the message reads "STILL BUSY" plus the time and date.

OUTPUTS:
none

COMMON BLOCKS:
BIZZY

SIDE EFFECTS:
The overhead of calling this function many times might be onerous; so don't bury it deep inside nested loops.

PROCEDURE:
Each time the procedure is invoked, it decodes the !STIME system variable to see if the minute has changed since the last time. If so, then the program prints the announcement. If not, it returns and does nothing.

EXAMPLE: The following code might be used in a coded procedure that runs for twenty minutes. About twenty times out of the 1000 passages through the loop, a message appears on the screen.

```
for i = 1, 1000 do begin
number_cruncher, data, i
WSTBUSY, "Processing iteration number " + strtrim(i,2)
endfor
```

MODIFICATION HISTORY:

Written by: James Hamill
Siemens Medical Systems
2501 N. Barrington Rd.
Hoffman Estates, IL 60195-7372
(708)304-7760
hamill@sgi.siemens.com
October, 1989
Renamed from still_busy.pro by Amara Graps for Wavelet Workbench, March 1996.

NAME:

WSUM

PURPOSE:

To sum all of the elements in an array or matrix. If A is a matrix, then wsum(A) is a row vector with the sum over each column. This function is an equivalent of Matlab's sum function.

CATEGORY:

Simple math.

CALLING SEQUENCE:

Result = WSUM(A)

INPUTS:

A: A vector or matrix

OUTPUTS:

The sum of the vector or matrix.

EXAMPLE:

```
IDL>a = [[8,2,6],[3,6,7],[4,9,3]]
IDL>PRINT, a
      8 2 6
      3 6 7
      4 9 3
IDL>PRINT, WSUM(A)
      15.0000      17.0000      16.0000
```

MODIFICATION HISTORY:

Amara Graps, BAER, December 1994.

NAME:
WSURETHR

PURPOSE:
To perform adaptive threshold selection using principle of Stein's Unbiased Risk Estimate (SURE)

CATEGORY:
Wavelets.

CALLING SEQUENCE:
x = WSURETHR(y, thresh)

INPUTS:
y = Noisy Data with Std. Deviation = 1

OUTPUTS:
thresh = Threshold used
x = Estimate of mean vector

EXAMPLE:

SEE ALSO:
multisure, wsoftthr, valwsurethr, wdoshrnk

MODIFICATION HISTORY:
Written by: Amara Graps September, 1995
(Translated from MatLab Wavelab routine: surethresh.m)

NAME:
WSWHERE

PURPOSE:
Same as 'where' function, except always returns a scalar, never a vector.

CATEGORY:
Arrays

CALLING SEQUENCE:
r = WWSWHERE(v)

INPUTS:
v = variable to use 'where' on

OUTPUTS:
r = return code
if where(v) is a scalar, r = where(v)
if where(v) is a vector, r = w(0), where w = where(v)

RESTRICTIONS:
Object and search_string must be of type string.

REQUIRED ROUTINES:
IDL Library: where, size, lonarr

MODIFICATION HISTORY:
T.Atwater 910510 – Created
Renamed from swhere.pro by Amara Graps for Wavelet Workbench, March 1996.

NAME:
WTREEHGT

PURPOSE:
To measure the total height of a stat-tree.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
maxheight = WTREEHGT(stree,D, cost)

INPUTS:
stree = stat tree (e.g. generated by WSTATREE)
D = maximum depth of search

OUTPUTS:
maxheight = Height of the tree, assigning each branch a length equal to the entropy drop between parent and child
cost = Tree of entropy drops associated with various splits

NOTES:
This is a utility used by WPLBTREE to set scale for plotting. It is not intended for other use.

EXAMPLE:

SEE ALSO:
wstatree, wplbtree

MODIFICATION HISTORY:
Written by: Amara Graps December 1994
(Translated from MatLab Wavelab routine: calctreeheight.m)

NAME:
WULKAXES

PURPOSE:
This procedure gets rid of the previously-set (xmin,xmax)(ymin,ymax) on the plot.

CATEGORY:
Plotting

CALLING SEQUENCE:
UNWLCKAXES

INPUTS:
None

OUTPUTS:
The x and y limits are set to automatic scaling

SEE ALSO:
wlckaxes

MODIFICATION HISTORY:
Written by: Amara Graps August, 1995
(Translated from MatLab Wavelab routine: UnlockAxes.m)

NAME:
WUPDYDHI

PURPOSE:
This function performs a Hi-Pass upsampling operation; periodized.

CATEGORY:
Wavelets

CALLING SEQUENCE:
y = WUPDYDHI(x,qmf)

INPUTS:
x: A 1-d signal at coarser scale.
qmf: quadrature filter

OUTPUTS:
y: A 1-d signal at finer scale

EXAMPLE:

SEE ALSO:
wdndydlo, wdndydhi, wupdydlo, wiwtpo, waconv

MODIFICATION HISTORY:
Written by: Amara Graps November, 1994
(Translated from MatLab Wavelab routine: updyadhi.m)

NAME:
WUPDYDLO

PURPOSE:
This function performs a Lo-Pass upsampling operation; periodized.

CATEGORY:
Wavelets

CALLING SEQUENCE:
u = WUPDYDLO(d,f)

INPUTS:
x: A 1-d signal at finer scale. qmf: quadrature mirror filter

OUTPUTS:
y: A 1-d signal at coarser scale

EXAMPLE:

SEE ALSO:
wdndydhi, wdndydlo, wupdydhi, wiwtpo, waconv

MODIFICATION HISTORY:
Written by: Amara Graps November, 1994
(Translated from MatLab Wavelab routine: updyadlo.m)

NAME:
WUPKCOEF

PURPOSE:
This function extracts the basis coefficients from packet table.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
x = WUPKCOEF(basis,wp)

INPUTS:
basis = basis tree
wp = packet table (wavelet)

OUTPUTS:
x = coefficients of signal in that basis

MODIFICATION HISTORY:
Written by: Amara Graps September, 1995
(Translated from MatLab Wavelab routine: unpackbasiscoeff.m)

NAME:
WUPSMPLE

PURPOSE:
This function performs an upsampling operation.

CATEGORY:
Wavelets

CALLING SEQUENCE:
y = WUPSMPLE(x)

INPUTS:
x: 1-d signal, of length n

OUTPUTS:
y: 1-d signal, of length 2**n with zeros interpolating alternate samples y(2*i-2) = x(i-1), i=1 ..n

EXAMPLE:

SEE ALSO:
wdndydhi, wdndydlo, wupdydhi, wiwtpo, waconv

MODIFICATION HISTORY:
Written by: Amara Graps November, 1994
(Translated from MatLab Wavelab routine: upsample.m)

NAME:
WVALSTHR

PURPOSE:
To perform adaptive threshold selection using principle of Stein's Unbiased Risk Estimate (SURE)

CATEGORY:
Wavelets.

CALLING SEQUENCE:
thresh = WVALSTHR(x)

INPUTS:
x = noisy data with std. deviation = 1

OUTPUTS:
thresh = value of threshold

EXAMPLE:

SEE ALSO:
wsoftthr, wsurethr, wdoshrnk

MODIFICATION HISTORY:
Written by: Amara Graps December, 1994
(Translated from MatLab Wavelab routine: valsurethresh.m)

NAME:
WVISUTHR

PURPOSE:
To perform visually calibrated adaptive smoothing.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
x = WVISUTHR(y, TThresh)

INPUTS:
y = data array
TThresh = type of Visu thresholding:
= 1 (Hard) or = 2 (Soft)

OUTPUTS:
x = data array thresholded

EXAMPLE:

SEE ALSO:
WHARDTHR, WSOFTTHR, WDOSHRNK

NOTES:
See article "Ideal Spatial Adaptation via Wavelet Shrinkage" By D.L. Donoho and I.M. Johnstone

MODIFICATION HISTORY:
Written by: Amara Graps September 1995
(Translated from MatLab Wavelab routine: visuthresh.m)

NAME:
WVLOADCT

PURPOSE:
This procedure sets up the color table and color variables used in the WaveLab plotting routines.

CATEGORY:
Wavelets

CALLING SEQUENCE:
WVLOADCT, col_table

INPUTS:
col_table: integer

OUTPUTS:
Color table and color variables set

EXAMPLE:

```
IDL>WVLOADCT, 1
(Loading table BLUE/WHITE)
```

NOTES:
Put the following common block in your program to use the pre-defined color variables (i.e., the call: PLOT, x, color = yellow should work).

```
COMMON Wave_color, $
      max_color, max_image, $
      white, yellow, lavender, aqua, pink, green, red, orange, blue, $
      lt_gray, med_green, brown, olive, purple, dk_gray, black
```

MODIFICATION HISTORY:
Written by: Amara Graps November, 1994
(lifted from IDL's tour_loadct.pro)

NAME:
WWB

PURPOSE:
This procedure sets up and runs the event loop for the Wavelet Workbench functions.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
WWB

INPUTS:
The user must specify what path the wavelets directory routines are located at. A query is made in an IDL widget when the user calls the wwb procedure.

OUTPUTS:
This procedure runs the wavelet data input as well as the actions performed on the data.

MODIFICATION HISTORY:
Written by: Amara Graps, August/September 1995.

NAME:
WWBDIR

PURPOSE:
This procedure sets up the location of the directory of Wavelet Workbench (WWB), so that the 1D and 2D data files can be found and loaded into the program menu.

CATEGORY:
Wavelets

CALLING SEQUENCE:
WWBDIR

INPUTS: (via COMMON)

OUTPUTS:
wd = working directory. This value is output in the form of a common block variable in WWB_COMMON

MODIFICATION HISTORY:
Written by: Amara Graps, September 1995.

NAME:
WWPLOC

PURPOSE:
This function calculates the location of a wavelet packet entry.

CATEGORY:
Wavelets.

CALLING SEQUENCE:
pos = WWPLOC(d,b,k,qmf,n)

INPUTS:
d = depth index
b = block index
k = coef index
qmf = qmf filter
n = signal length

OUTPUTS:
pos = position in wavelet packet data structure

MODIFICATION HISTORY:
Written by: Amara Graps November, 1995
(Translated from MatLab Wavelab routine: calcwplocation.m)

1. Wavelet Workbench Origins and Oddities

1.1. Origins

The core of the WWB library are functions and scripts provided with permission by the authors of the Matlab WaveLab software. WaveLab is the basis for wavelet research by the Stanford, Berkeley and NASA-Ames authors, as well as used in teaching wavelet and digital signal processing courses.

1.2. Oddities

I (Amara) have faithfully reproduced many aspects of the core routines of the WaveLab software, which means that, if any odd behavior exists in the WaveLab software, it has now made into this Wavelet Workbench library. The (unfatal) behavior don't seem to affect the results, therefore I'm keeping these routines as they are originally written and I'm documenting them in this page for future reference.

SCALEGRAM
The scalegram currently uses *only* the Haar wavelet. The WaveLab authors feel that, in general, it is the only one that you can use to get down to the "finest scale." With the Haar wavelet there are no worries about end-effects (i.e. no wavelet sticking out over the ends of the data). Plus, the correction of the resulting scalegram using the Haar wavelet has a very simple form (= mean counting rate) for Poisson statistics in the data.

DENOISING
The `WVISUTHR` and `WMTHRESH` algorithms use input array lengths that are not dyadic length (i.e. array length = 2**J long). You will see printed error: 'Warning in dyadlength: n is not equal to 2**J', but your results will be fine.

2D DATA (IMAGES)
The 2D Data provided in Wavelet Workbench were taken directly from the Matlab WaveLab software. The reading and writing to save these images in IDL introduced some artifacts. I would really like to have some better images! If you have some nice images that you'd be willing to donate to Wavelet Workbench, please contact me (Amara, aka `agraps@netcom.com`).

2. Wavelet Workbench Future Development

The Wavelet Workbench development could go on forever because the wavelet field is vast, as is the WaveLab software. However, it is necessary to get what is done out and in the hands of the users, so here is my (Amara) list of future development for Wavelet Workbench.

2D WAVELET FUNCTIONS
More 2D wavelet functions will be added, including 2D denoising and wavelet packet analysis.

2D DWT DISPLAY
The resulting coefficient display from the Discrete Wavelet Transform of a 2D dataset is low resolution (due to the values of the coefficients being so tiny) and the displayed values need considerable stretching. Future work will make this image display more visible.

SCALOGRAM
The scalogram calculated and displayed is simply a discrete wavelet transform in a surface representation. I don't feel that any new information is presented in this sort of choppy display. A better wavelet transform for the scalegram would be the continuous wavelet transform, and that is my goal for a future version.

OPTIMIZATION AND SPEED
For the initial creation of Wavelet Workbench, my effort focussed on making the results accurate and imitating the functionality of WaveLab as much as possible. The drawback is that the initial version of Wavelet Workbench may be slow – no attention was given to the optimization. Future versions will focus on speeding up Wavelet Workbench, in particular, the 2D functions. It may even become necessary to call C or Fortran routines from IDL to achieve these speed gains.

This Wavelet Workbench manual is not designed to teach you wavelet concepts, but instead, to describe how to use the software. If you are finding that you'd like to study up on the subject of wavelets, the following references are selected for ease of understanding by someone unfamiliar with the signal processing literature. These wavelet references should help you understand various concepts in the Wavelet Workbench. The easiest are listed first.

(1) Cipra, Barry A, "Wavelet Applications Come to the Fore," *SIAM News (Mathematics that Counts,* 11/93. gopher://gopher.siam.org:70/00/siamnews/mtc/mtc1193.txt

(2) Graps, A.L., "An Introduction to Wavelets," *IEEE Computational Sciences and Engineering,* Summer 1995. ftp://ftp.amara.com/papers/IEEEwavelet.ps.gz

(3) Strang, G., "Wavelets," *American Scientist,* Vol. 82, 1994, pp. 250-255.

(4) Meyer, Yves, "Review of two books on wavelets," *Bull. (New Series) Amer. Math. Soc.,* Vol. 28, No. 2, 1993, pp. 350-360.

(5) Meyer, Yves, *Wavelets: Algorithms and Applications,* Society for Industrial and Applied Mathematics, Philadelphia, 1993.

(6) Rioul, Olivier and Martin Vetterli, "Wavelets and Signal Processing," *IEEE Signal Processing Magazine,* October 1991, p.14-38.

(7) Strang, G., "Wavelet transforms versus Fourier transforms," *Bull. (New Series) Amer. Math. Soc.,* Vol. 28, No. 2, 1993, pp. 288-305.

(8) Robinson, Sam L. and Ryczek, Peter F., "Wavelets," *The Mathematica Journal,* Volume 5, Issue 1, Winter 1995.

(9) W. Press et al., *Numerical Recipes in Fortran,* Cambridge University Press, New York, 1992, pp. 498-499, 584-602.

(10) Chan, T., *Wavelet Basics,* Kluwer Academic Publishers, 1995.

(11) M. Vetterli and C. Herley, "Wavelets and Filter Banks: Theory and Design," *IEEE Transactions on Signal Processing,* Vol. 40, 1992, pp. 2207-2232.

(12) R. Crandall, *Projects in Scientific Computation,* Springer-Verlag, New York, 1994, pp. 197-198, 211-212.

(13) I. Daubechies, "Orthonormal Bases of Compactly Supported Wavelets," *Comm. Pure Appl. Math.,* Vol 41, 1988, pp. 906-966.

(14) Daubechies, Ingrid, *Ten Lectures on Wavelets,* Society for Industrial and Applied Mathematics Press, vol. 61 of CBMS-NSF Regional Conference Series in Applied Mathematics, Philadelphia, 1992.

(15) V. Wickerhauser, *Adapted Wavelet Analysis from Theory to Software,* AK Peters, Boston, 1994.

(16) Martin Vetterli and Jelena Kovacevic, *Wavelets and Subband Coding,* Prentice-Hall, New Jersey, 1995.

(17) Newland, D.E., *An Introduction to Random Vibrations, Spectral and Wavelet Analysis,* New York, John Wiley, 1993.

(18) Young, R.K., *Wavelet Theory and its Applications,* Kluwer Academic Publishers, ISBN 0-7923-9271-X, 1993.

(19) M.A. Cody, "The Fast Wavelet Transform," *Dr. Dobb's Journal,* Apr. 1992.

(20) M.A. Cody, "A Wavelet Analyzer," *Dr. Dobb's Journal,* Apr. 1993.

(21) M.A. Cody, "The Wavelet Packet Transform," *Dr. Dobb's Journal,* Apr. 1994, pp. 44-46, 50-54.

(22) Mallat, S.G., "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol II, 7, 1989, pp. 674-693.

(23) J. Bradley, C. Brislawn, and T. Hopper, "The FBI Wavelet/Scalar Quantization Standard for Gray-scale Fingerprint Image Compression," Tech. Report LA-UR-93-1659, Los Alamos Nat'l Lab, Los Alamos, N.M. 1993.

(24) D. Donoho, "Nonlinear Wavelet Methods for Recovery of Signals, Densities, and Spectra from Indirect and Noisy Data," *Different Perspectives on Wavelets, Proceeding of Symposia in Applied Mathematics,* Vol 47, I. Daubechies ed. Amer. Math. Soc., Providence, R.I., 1993, pp. 173-205.

(25) B. Vidakovic and P. Muller, "Wavelets for Kids," 1994, unpublished. Available by FTP at `ftp://ftp.isds.duke.edu` in directory `/pub/brani/papers/wav4kids[A-B].ps.Z`

(26) J. Scargle et al., "The Quasi-Periodic Oscillations and Very Low Frequency Noise of Scorpius X-1 as Transient Chaos: A Dripping Handrail?," *Astrophysical Journal,* Vol. 411, 1993, L91-L94.

(27) M.V. Wickerhauser, "Acoustic Signal Compression with Wave Packets," 1989. Available by anonymous FTP at `pascal.math.yale.edu`, filename: `acoustic.tex`.

(28) Ali N. Akansu and Richard A. Haddad, *Multiresolution Signal Decomposition Transforms, Subbands, Wavelets,* Academic Press, Inc., ISBN 0-12-047140-X.

(29) Weiss, L. G., "Wavelets and Wideband Correlation Processing," *IEEE Transactions on Signal Processing,* January 1994, p13-32.

(30) G. Kaiser, *A Friendly Guide to Wavelets,* Birkhauser, Boston, 1994.